

Copyright

by

Ye Zhang

2019

**The Dissertation Committee for Ye Zhang
certifies that this is the approved version of the following dissertation:**

Neural NLP Models Under Low-supervision Scenarios

Committee:

Matthew A Lease, Supervisor

Byron Wallace, Co-Supervisor

Raymond J Mooney

Gregory C Durrett

Neural NLP Models Under Low-supervision Scenarios

by

Ye Zhang

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2019

Neural NLP Models Under Low-supervision Scenarios

Ye Zhang, Ph.D.

The University of Texas at Austin, 2019

Supervisors: Matthew A Lease and Byron Wallace

Neural models have been shown to work well for natural language processing tasks when one has large amounts of labeled data, but problems arise when this is not the case. In this thesis we investigate several ‘low-supervision’ scenarios in which we do not have sufficient training data, and we propose methods to improve performance in these scenarios.

First, we consider the scenario where we can use other types of resources in addition to the limited training labels. For instance, we can ask human annotators to provide *rationales* supporting their labels (annotations) for training examples. To capitalize on such supervision, we develop a neural model that can train on both instance labels and associated rationales. We also investigate how to incorporate existing *ontologies* into neural models. Specifically, we develop a novel training algorithm that enforces weight sharing among similar words in the ontologies, thus inductively biasing the neural model training.

In addition incorporating other types of resources beyond instance labels, we also use transfer learning techniques which are general means of learning in low-supervision settings. We study how to use multiple sets of pre-trained word embeddings as inputs to neural models, and fine-tune them to the task at hand in a more intelligent way than simply concatenating them. We also develop a novel model for text generation, in which the model is able to generate text from a new domain (unseen in training data). Rather than simply fine-tuning the model on the target domain, the model fully uses the domain information in the training set, allowing it to generate domain specific text.

Lastly, we consider how to collect data under a limited budget more effi-

ciently than simply random selection of unlabeled data for annotation. We develop new active learning (AL) methods to collect more informative examples to be annotated specifically for neural models, so that better models and more discriminative text representation can be learned with fewer labels. Following this, we further develop new AL approaches when we have richly annotated data from a relevant domain, that is, we combine AL and transfer learning and leverage the advantages of both methods. We also investigate how to use the pre-trained deep bidirectional transformer (BERT) to actively select labels.

Table of Contents

Chapter 1	Introduction	1
1.1	Thesis Contributions	3
Chapter 2	Background	6
2.1	Convolutional Neural Networks	6
2.2	Recurrent Neural Networks	7
2.2.1	LSTM	9
2.2.2	GRU	10
2.2.3	Seq2seq	11
2.3	Low Supervision Scenarios	11
Chapter 3	Neural Models Augmented with Human Rationales	13
3.1	Chapter Overview	13
3.2	Prior Work	14
3.2.1	Exploiting <i>rationales</i>	14
3.3	Rationale Augmented Neural Models	15
3.3.1	Modeling Document Structure	15
3.3.2	RA-CNN	15
3.3.3	Rationales as ‘Supervised Attention’	19
3.4	Datasets	19
3.4.1	Risk of Bias (RoB) Datasets	20
3.4.2	Movie Review Dataset	20
3.5	Experimental Setup	21
3.5.1	Baselines	21
3.5.2	Implementation/Hyper-Parameter Details	22
3.6	Results and Discussion	24
3.6.1	Quantitative Results	24
3.6.2	Qualitative Results: Illustrative Rationales	25
3.7	Chapter Summary	26

Chapter 4	Neural Models Augmented with Ontologies	27
4.1	Chapter Overview	27
4.2	Prior Work	28
4.3	Weight Sharing in Embedding Space	29
4.4	Experimental Setup.....	33
4.4.1	Datasets.....	33
4.4.2	Implementation Details and Baselines	33
4.5	Results.....	35
4.6	Chapter Summary	36
Chapter 5	Neural Models Augmented with Multiple Pre-trained Embeddings	37
5.1	Chapter Overview	37
5.2	Prior Work	38
5.3	Method	39
5.4	Experiments	42
5.4.1	Datasets.....	42
5.4.2	Pre-trained Word Embeddings	42
5.4.3	Setup.....	43
5.4.4	Results and Discussion	44
5.5	Chapter Summary	45
Chapter 6	Domain Adaptation for Neural Text Generation.....	46
6.1	Chapter Overview	46
6.2	Prior Work	49
6.3	Shared-Private Encoder Decoder Model for Sequence Generation	51
6.3.1	SHAPED: Shared-private encoder-decoder	51
6.3.2	The Mix-SHAPED Model	54
6.3.3	Model Instantiation.....	56
6.3.4	SHAPED Instantiation	56
6.4	Experiments	57
6.4.1	Experimental Setup	58
6.4.2	Main Results.....	60

6.4.3	Experiment Variants	61
6.5	Chapter Summary	64
Chapter 7	Active Discriminative Text Representation Learning	65
7.1	Chapter Overview	65
7.2	Prior Work	66
7.3	Method	68
7.3.1	Active Sentence Classification with CNNs	69
7.3.2	Active Document Classification with CNNs	70
7.4	Experiment	72
7.4.1	Model Configuration	73
7.5	Results and Discussion	74
7.5.1	Sentence Classification Results	75
7.5.2	Document classification results	77
7.6	Chapter Summary	78
Chapter 8	Active Transfer Learning for Neural Models	79
8.1	Chapter Overview	79
8.2	Prior Work	80
8.3	Influence Function and Explaining Predictions in NLP	80
8.3.1	Text Classification	82
8.3.2	Sequence Tagging	84
8.4	Active Transfer Learning via the Influence Function	88
8.5	Experiments	91
8.5.1	Active Transfer: Text Classification	92
8.5.2	Active Transfer: Sequence Tagging	94
8.5.3	Active Transfer: Multi-Genre Natural Language Inference	95
8.5.4	Failure analysis and possible improvement	98
8.6	Chapter Summary	101
Chapter 9	Active Learning with Pre-trained BERT	102
9.1	Chapter Overview	102

9.2	Prior Work	102
9.3	AL with BERT (Active BERT)	103
9.3.1	Preliminaries on BERT	103
9.3.2	BERT for Active Learning	104
9.4	Experiments	105
9.5	Results.....	107
9.6	Comparison with AL methods in Chapter 7 and Chapter 8.....	108
9.7	Chapter Summary	109
Chapter 10	Conclusion and future work.....	110
10.1	Future Work	111
10.1.1	Rationales in Active Learning.....	111
10.1.2	Rationales for Structured Neural Models	112
10.1.3	Active Learning for Structured Neural Models	112
References	113
Vita	129

List of Tables

3.1	Dataset characteristics. N is the number of instances, $\#sen$ is the average sentence count, $\#token$ is the average token per-sentence count and $\#rat$ is the average number of rationales per document.	21
3.2	Accuracies on the four RoB datasets. Uni-SVM: unigram SVM, Bi-SVM: Bigram SVM, RA-SVM: Rationale-augmented SVM (Zaidan et al., 2007), MT-SVM: a multi-task SVM model specifically designed for the RoB task, which also exploits the available sentence supervision (Marshall et al., 2016). We also report an estimate of human-level performance, as calculated using subsets of the data for each domain that were assessed by two experts (one was arbitrarily assumed to be correct). We report these numbers for reference; they are not directly comparable to the cross-fold estimates reported for the models.....	23
3.3	Accuracies on the movie review dataset.	25
4.1	Corpora statistics.	33
4.2	Accuracy mean (min, max) on sentiment datasets. ‘p’: channel initialized with the pre-trained embeddings \mathbf{E}^p . ‘r’: channel randomly initialized. ‘retro’: initialized with retofitted embeddings. ‘S/B (no sharing)’: channel initialized with \mathbf{E}^s (using SentiWordNet or Brown clusters), but weights are not shared during training. ‘S/B (sharing)’: proposed weight-sharing method.	36

4.3	AUC mean (min, max) on the biomedical datasets. Abbreviations are as in Table 4.2, except here the external resource is the UMLS MeSH ontology ('U'). 'U(s)' is the proposed weight sharing method utilizing ULMS.	36
5.1	Results mean (min, max) achieved with each method. w2v:word2vec. Glv:GloVe. Syn: Syntactic embedding. Note that we experiment with using two and three sets of embeddings jointly, e.g., w2v+Syn+Glv indicates that we use all three of these.....	41
5.2	Best λ^2 value on the validation set for each method w2v:word2vec. Glv:GloVe. Syn: Syntactic embedding.....	41
6.1	ROUGE F1 scores on the combined AFP/APW/XIN/NYT in-domain test set.	61
6.2	ROUGE F1 scores on out-of-domain style test sets CNA and LTW. ..	61
7.1	Statistics of sentence datasets.....	72
7.2	Statistics of document datasets. l_{doc} denotes the average sentence length in words, and l_{doc} denotes the average number of sentences per document.....	72
7.3	Area Under (learning) Curve (AUC) scores on sentence classification datasets; bold indicates best results.	77
7.4	Area Under (learning) Curves (AUC) scores on the three document datasets. E-E-B refers to <i>EGL-Entropy-Beta</i>	77
8.1	Sentence count in the movie review (MR) sentiment dataset, and test accuracy (%) achieved by AVG and LSTM models on it.....	82
8.2	A misclassified movie review (MR) sentence and corresponding 'most harmful' training point, as identified via the influence function.	83
8.3	Number of sentences in the train, dev and test splits of the CoNLL 2003 and Twitter NER corpora used in sequence tagging experiments.	84

8.4	Three selected test entities in the CoNLL 2003 NER dataset and corresponding most harmful training entities identified by the influence function. For each test entity, we report both its predicted and ‘true’ label (\hat{y} , y). The first two examples show a test entity tagged incorrectly by the model while the last example shows a test entity tagged correctly. For readability, we have normalized casing.	86
8.5	Area Under learning Curves for sentence classification achieved using active transfer methods with LSTM (corresponding to Fig. 8.3)...	95
8.6	Area under learning curves for sentence classification achieved using active transfer methods with CNN (corresponding to Fig. 8.5)....	95
8.7	AUCs for sequence tagging (NER) achieved by different active transfer learning strategies (corresponding to Fig. 8.6). ‘C to T’ denotes transferring from CoNLL to Twitter, while ‘T to C’ denotes the reverse.	96
8.8	AUCs for MultiNLI achieved by different active transfer learning strategies (corresponding to Fig. 8.7). ‘G’ is government. ‘T’ is travel. ‘F’ is fiction. ‘TE’ is telephone	99
9.1	Statistics of the classification (top) and sequence tagging (bottom) corpora used for experiments. #train denotes the initial total number of unlabeled instances in the pool, #eval the number we evaluate on.	106
9.2	AUC scores averaged over five runs for each method on each dataset. ‘ran’ is random sampling, ‘ent’ is entropy, ‘per’ is perplexity, ‘ent+per’ is entropy + perplexity. Corpora above the horizontal line are text classification tasks; below are sequence tagging. ‘EBM-B’ denotes Bio-BERT initialization on EBM, and ‘EBM-V’ denotes the standard version.	108

List of Figures

2.1	Illustration of a CNN architecture for sentence classification. We depict three filter region sizes: 2, 3 and 4, each of which has 2 filters. Filters perform convolutions on the sentence matrix and generate (variable-length) feature maps; 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we assume binary classification and hence depict two possible output states.	8
2.2	Recurrent Neural Network.....	10
2.3	Sequence to Sequence Network	11
3.1	A schematic of our proposed Rationale-Augmented Convolution Neural Network (RA-CNN). The sentences comprising a text are passed through a sentence model that outputs probabilities encoding the likelihood that sentences are neutral or a (positive or negative) rationale. Sentences likely to be rationales are given higher weights in the global document vector, which is the input to the document model.	16
4.1	An example of grouped partial weight sharing. Here there are two groups. We stochastically select embedding weights to be shared between words belonging to the same group(s).....	28

4.2	Proposed two-channel model. The first channel input is a standard pre-trained embedding matrix. The second channel receives a partially shared embedding matrix constructed using external linguistic resources.....	31
5.1	Illustration of MG-CNN and MGNC-CNN. The filters applied to the respective embeddings are completely independent. MG-CNN applies a max norm constraint to \mathbf{o} , while MGNC-CNN applies max norm constraints on \mathbf{o}_1 and \mathbf{o}_2 independently (group regularization). Note that one may easily extend the approach to handle more than two embeddings at once.	40
6.1	Illustration of the SHAPED model using two styles D_1 and D_2 . D_1 articles pass through the private encoder f_{enc}^1 and decoder f_{dec}^1 . D_2 articles pass through the private encoder f_{enc}^2 and decoder f_{dec}^2 . Both of them also go through the shared encoder f_{enc}^s and decoder f_{dec}^s	52
6.2	Decoding data with unknown style using a Mix-SHAPED model. The data is run through all encoders and decoders. The output of private encoders is fed into a classifier that estimates style distribution. The output symbol distribution is a mixture over all decoder outputs.	55
6.3	(a) Each example is fed to all private encoders f_{enc}^1, f_{enc}^2 , whose outputs are concatenated and fed to a style classifier. (b) The D_1 examples only use $f_{enc}^1, f_{dec}^1, f_{enc}^s, f_{dec}^s$ to decode texts. Private encoder-decoders of other styles are not used.	56
6.4	Experimental results on the headline generation task, for in-domain styles.	60
6.5	Estimated style probabilities over the four in-domain styles, for out-of-domain styles CNA and LTW.....	62
7.1	Beta distributions over γ_t at $t=0, t=10, t=20$	73

7.2	Results on the three sentence classification datasets. Top row: number of labels versus accuracy. Bottom row: number of labels versus the distance between tuned embeddings for selected pairs of informative words (with opposite polarity) for each dataset. The scale in this case, which captures the Euclidean distance in the embedding space, has only relative meaning.	75
7.3	Results on the three document datasets. Top row: number of labels versus accuracy. Bottom row: number of labels versus the distance between tuned embeddings for selected pairs of informative words (with opposite polarity) in each task. Methods that explicitly consider representation/embedding parameters more quickly push discriminative word vectors apart. Intuitively, the distances between the contrasting word-pairs increases quickly with both of the proposed EGL methods. However, recall that the <i>EGL-Entropy-Beta</i> method differs from <i>EGL-word-doc</i> in interpolating entropy along with expected updates to word gradients. As a result, we observe that <i>EGL-Entropy-Beta</i> method tends to shift from rising with <i>EGL-word-doc</i> at the start of learning, while later merging with the distances achieved by the <i>Entropy</i> method as learning progresses. This transition corresponds to first focusing on embeddings, and then later shifting emphasis to the entropy criterion.....	76
8.1	The relationship between influence function values and distances between embedded instances. For test point z_{test} , we show a scatter plot of the influence function of each training point w.r.t. z_{test} against the Euclidean distances between these. Blue \times 's and yellow Δ 's denote positive and negative training points, respectively.	85
8.2	Active transfer via the influence function. Labels in \mathcal{S} are known; those in \mathcal{T} are not.	89

8.3	Target domain accuracy (y-axis) of transfer learning for document classification using LSTM with varying annotation budget (x-axis) for the influence function approach ('Inf') vs. 3 baselines.....	93
8.4	Target domain accuracy (y-axis) of transfer learning for short sentence classification using LSTM with varying annotation budget (x-axis) for the influence function approach ('Inf') vs. 3 baselines.....	94
8.5	Target domain accuracy (y-axis) of active transfer learning for text classification (i.e., sentiment analysis) using CNN with varying annotation budget (x-axis) for the influence function approach ('Inf') vs. 3 baselines.	96
8.6	Target domain F1-score (y-axis) of transfer learning for sequence tagging (i.e., NER) with varying annotation budget (x-axis) for the influence function approach ('Inf') vs. the 3 baselines.....	97
8.7	Active transfer learning curve for MultiNLI. y-axis is target domain F1-score and x-axis is the number of labeled target instances.	98
9.1	Active Learning Curve for classification dataset.....	107
9.2	Active Learning Curve for NER dataset.	107

Chapter 1

Introduction

Neural models are powerful when there is large scale training data, e.g., there may be millions of customer product reviews that Amazon can use to train a sentiment classifier (Kim, 2014; Zhang and Wallace, 2015a). However, it is not often the case that we have access to that amount of training data because annotating data is time-consuming and expensive. Prior works have proposed multiple methods to handle ‘low-supervision’ scenarios where we don’t have large training data. Some of the most commonly used strategies include data augmentation (Vincent et al., 2008), semi-supervised learning (Grandvalet and Bengio, 2005), training beyond instance labels (Faruqui et al., 2014), multi-task learning (Collobert and Weston, 2008), regularization (Hinton et al., 2012), parameter sharing (Han et al., 2015), unsupervised learning (Mikolov et al., 2013b), transfer learning (Mou et al., 2016), and active learning (Settles, 2010). This thesis focuses on three of the above strategies: training beyond instance labels, transfer learning and active learning. For each of the strategies, we propose novel methods specifically designed for neural NLP models to improve model performance under low-supervision scenarios.

The classical way of doing supervised machine learning is using training instance labels. For example, when doing sentiment analysis, people build models that rely on instances and their sentiment scores (Zhang and Wallace, 2015a). But we go beyond instance labels for low-supervision scenarios. Instead of just using instance labels, we also incorporate other types of resources into neural model training. In this thesis we consider augmenting neural models with human *rationales* (Zhang et al., 2016a) and *ontologies* such as wordnet (Baccianella et al., 2010). In Chapter 3, we consider the scenario where we not only ask human annotators to label training instances, but also to provide further rationales that explain their annotations. These rationales can augment the limited training labels. Prior works have incorporated rationales into support vector machines (SVMs) (Zaidan et al., 2007). In this thesis, we propose a neural model that can train on both in-

stance labels and rationale labels. In Chapter 4, we consider how to incorporate ontologies (Zhang et al., 2017b) as prior domain knowledge into neural models. We propose a novel training method that forces similar words in ontologies to share more parameters in the embedding space to inductively bias the neural model training.

In addition to augmenting models with other types of resources beyond instance labels, another way of augmenting models is transfer learning (Pan and Yang, 2010), where there is not enough data for the target task, but there is large scale training data for another relevant source task. Typically, we can first train a model on the richly annotated source task, and then transfer the trained source model to the low-supervision target task. This is typically better than directly training a model on the target task (Mou et al., 2016). In transfer learning, it is often the case that the source data contains a mixture of diverse attributes. For example, there might have several off-the-shelf pre-trained embeddings trained on different corpus using different training methods, or the source data might contain multiple various written styles. Prior works often ignore this issue (Rush et al., 2015; See et al., 2017). But the fact is that some part of the source data is beneficial for the transfer, but some other part might hurt the transfer due to the very different properties between the source task and the target task. Instead, we develop novel transfer learning methods that better leverage multiple types of source data to better suit the source task to the target task. In Chapter 5, we study how to better leverage multiple sets of pre-trained word embeddings in neural models and to cleverly fine-tune them to downstream tasks. In Chapter 6, we study text generation when various written styles are available in the training data and how to generate target style text using the mixture of source styles. We introduce a novel encoder decoder that is able to generate text sequences with any specific style even if the style has little training data or is totally unseen in the training set.

Lastly, we consider active learning (AL) for low-supervision scenarios. AL is more straightforward than the previous two methods for low-supervision scenarios, because it directly involves data collection. In AL, the learning algorithm is allowed to choose more valuable unlabeled instances to be annotated. The goal is

to achieve good performance with as few labeled instances as possible. People have developed AL approaches for traditional machine learning models (Settles, 2010). But neural NLP models are different than traditional machine learning models in that they feature a representation learning process (Bengio, 2009). So it is crucial that for neural models, we should introduce a better representation layer as the input to higher layers. In Chapter 7, we develop a new AL method specifically for neural NLP models that encourages more discriminative word embedding learning. Although there is an increasing attention on ALs for neural models recently (Siddhant and Lipton, 2018), they all do ALs from scratch. Instead, we also consider combining AL and transfer learning. We consider a situation where we can collect a limited amount of labels for the target domain, but at the same time, we can also pre-train a good model from a richly annotated relevant source domain. The goal is to train a good model on the target domain. In Chapter 8, we investigate how to actively choose target instances given a pre-trained source model as the initialization point. Other than initializing the target model from a relevant source domain, in Chapter 9, we also study how to do active learning on top of a pre-trained deep language model such as pre-trained deep bidirectional transformers (BERT).

1.1 Thesis Contributions

Training beyond instance labels

We investigate how to train neural models beyond instance labels and incorporate other types of resources into neural model training.

We propose a novel training method that can incorporate ontologies into neural model training. Ontologies as prior domain knowledge can teach the model which words should be semantically similar, so that their parameters should be shared in the embedding space. Parameter sharing is a commonly used regularization method for neural models, but it has been mainly used in computer vision as way of model compression (Chen et al., 2015; Han et al., 2015). We are the first to use domain knowledge to inductively bias the parameter sharing for neural NLP models.

We propose a novel neural model augmented with human rationales. We are the first to incorporate rationales into neural models. Rationale-augmented neural models can not only achieve strong predictive performance, but also provide explanations for their predictions at test time. Explainability is an important issue with neural models, and we are the very early one studying this issue.

Transfer learning for multiple types of source data

Instead of transfer learning from a single source, we consider transfer learning from multiple types of source data. Simply mixing all the source data together would ignore their diverse properties and their differences from the target. We develop new methods that can better leverage various types of source data.

We propose a new fine-tuning approach that can better leverage multiple pre-trained word embeddings. More specifically, we put independent regularization strengths on each embedding. We show that this method works better than simply mixing different embeddings.

We propose a new domain adaptation method for text sequence generation. Rather than training a single model on the training data which contains various written styles, our method can better separate different source styles and accordingly generate better target appropriate style text.

Active Learning for neural models

Although there is much work on AL methods for general machine learning models, we are the first one to develop AL methods specifically for representation learning in neural NLP models. Representation learning is a unique process for neural NLP models. Accordingly, our AL approach encourages more discriminative feature representation learning, so that higher layers can have better inputs. This is fundamentally different than traditional ALs which might already have discriminative features at hand. We show that our AL approach can induce more discriminative word representations and thus results in higher accuracy.

Instead of doing AL from scratch, we also consider combining AL with

transfer learning. We consider a scenario where we initialize the target model with a pre-trained source model, and we propose a novel AL approach that selects target instances that are maximally misaligned with source instances. We propose a novel AL method on top of pre-trained deep language models. The method uses perplexity of the pre-trained model on each unlabeled instance for data selection. We show that AL on top of pre-trained deep language models can provide further gains than using pre-trained models alone.

Chapter 2

Background

In this section, we briefly review the basic neural models (Onal et al., 2018) that we will use in later chapters, and then we illustrate the low supervision scenarios.

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) have been shown to work well for text classification. We will use them extensively in the thesis, so in this section, we elaborate on their details. A tokenized sentence can be converted to a *sentence matrix*, the rows of which are word vectors for each token. These might be, e.g., outputs from trained word2vec (Mikolov et al., 2013b) or GloVe (Pennington et al., 2014) models. We denote the dimensionality of the word vectors by d . If the length of a given sentence is s , then the dimensionality of the sentence matrix is $s \times d$.¹ Following Collobert and Weston (2008), we can then effectively treat the sentence matrix as an ‘image’, and perform convolution on it via linear *filters*. In text applications there is inherent sequential structure to the data. Because rows represent discrete symbols (namely, words), it is reasonable to use filters with widths equal to the dimensionality of the word vectors (i.e., d). Thus we can simply vary the ‘height’ of the filter, i.e., the number of adjacent rows considered jointly. We will refer to the height of the filter as its *region size*.

Suppose that there is a filter parameterized by the weight vector \mathbf{w} with region size h ; \mathbf{w} will contain $h \cdot d$ parameters to be estimated. We denote the sentence matrix by $\mathbf{A} \in \mathbb{R}^{s \times d}$, and use $\mathbf{A}[i : j]$ to represent the sub-matrix of \mathbf{A} from row i to row j . The output sequence $\mathbf{o} \in \mathbb{R}^{s-h+1}$ of the convolution operator

¹We use the same zero-padding strategy as in (Kim, 2014).

is obtained by repeatedly applying the filter on sub-matrices of \mathbf{A} :

$$o_i = \mathbf{w} \cdot \mathbf{A}[i : i + h - 1], \quad (2.1)$$

where $i = 1 \dots s - h + 1$, and \cdot is the dot product between the sub-matrix and the filter (a sum over element-wise multiplications). We add a bias term $b \in \mathbb{R}$ and an activation function f to each o_i , inducing the *feature map* $\mathbf{c} \in \mathbb{R}^{s-h+1}$ for this filter:

$$c_i = f(o_i + b). \quad (2.2)$$

One may use multiple filters for the same region size to learn complementary features from the same regions. One may also specify multiple kinds of filters with different region sizes (i.e., ‘heights’).

The dimensionality of the feature map generated by each filter will vary as a function of the sentence length and the filter region size. A pooling function is thus applied to each feature map to induce a fixed-length vector. A common strategy is *1-max pooling* (Boureau et al., 2010), which extracts a scalar from each feature map. Together, the outputs generated from each filter map can be concatenated into a fixed-length, ‘top-level’ feature vector, which is then fed through a softmax function to generate the final classification. At this softmax layer, one may apply ‘dropout’ (Hinton et al., 2012) as a means of regularization. This entails randomly setting values in the weight vector to 0. One may also impose an l_2 norm constraint, i.e., linearly scale the l_2 norm of the vector to a pre-specified threshold when it exceeds this. Figure 2.1 provides a schematic illustrating the model architecture just described.

For a sensitivity analysis of this CNN architecture and associated hyper-parameters, we refer the reader to (Zhang and Wallace, 2015a).

2.2 Recurrent Neural Networks

Unlike CNNs, Recurrent Neural Networks (RNN) are specialized to processing sequential data. They read input symbols step by step, and generate output

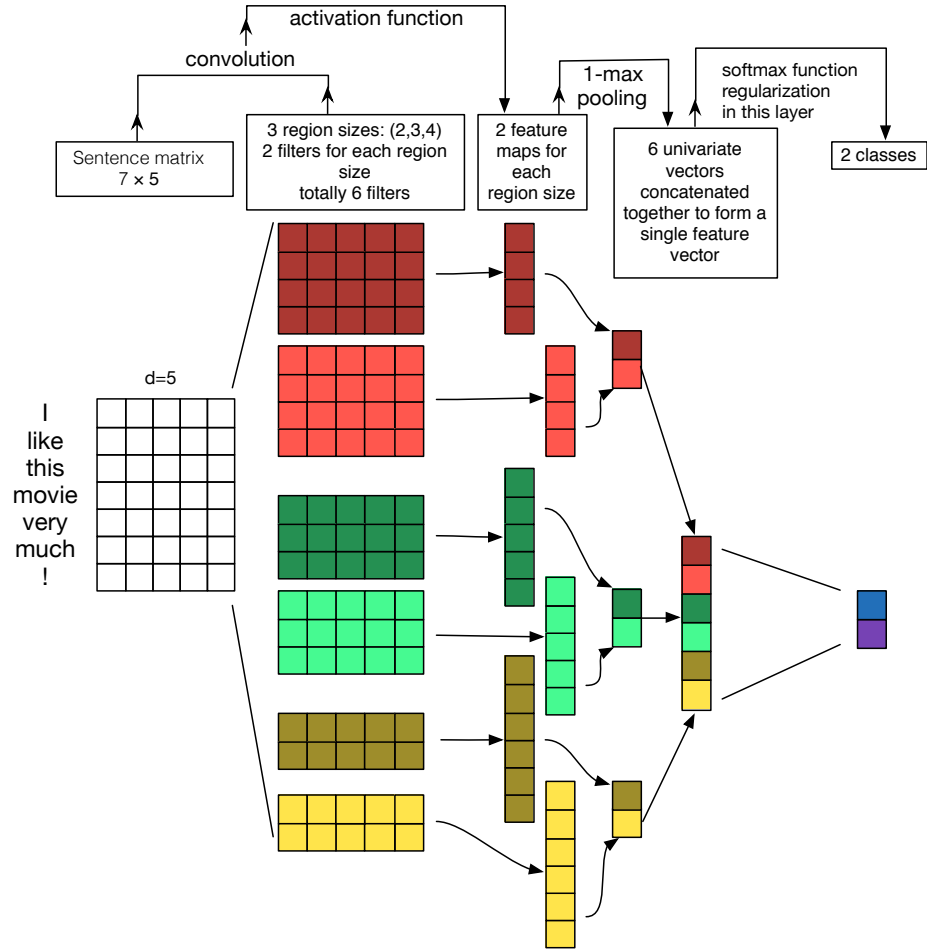


Figure 2.1: Illustration of a CNN architecture for sentence classification. We depict three filter region sizes: 2, 3 and 4, each of which has 2 filters. Filters perform convolutions on the sentence matrix and generate (variable-length) feature maps; 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we assume binary classification and hence depict two possible output states.

accordingly at each time step:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) \quad (2.3)$$

where x is the input, and h denotes a hidden state. The output is predicted using h .

We depict a simple RNN in Figure 2.2. The problem of RNNs is that gradient propagated over many steps tend to vanish or explode, meaning that the magnitude of the gradients become exponentially small or large, which makes it hard to capture long dependencies sometimes (Pascanu et al., 2013). To overcome these issues, people have developed various more sophisticated versions of RNNs. Two popular variants that are widely used and have been shown to work well on different tasks are Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks.

2.2.1 LSTM

LSTM is designed to allow the gradient to flow back for long durations by introducing a self-loop on the so the called cell state S . The self-loop is controlled by a forget gate:

$$f^t = \sigma(W_f h^{t-1} + U_f x^t + b_f) \quad (2.4)$$

where σ is sigmoid function. Which cell state value will be updated is controlled by an input gate:

$$i^t = \sigma(W_i h^{t-1} + U_i x^t + b_i) \quad (2.5)$$

Then a new candidate cell state is calculated as:

$$\hat{S}^t = \tanh(W_S h^{t-1} + W_S x^t + b_S) \quad (2.6)$$

The new cell state is the combination between the candidate cell state and the old cell state, controlled by the forget gate and the input gate:

$$S^t = f^t S^{t-1} + i^t \hat{S}^t \quad (2.7)$$

The hidden state or output vector is controlled by an output gate:

$$o^t = \sigma(W_o h^{t-1} + U_o x^t + b_o) \quad (2.8)$$

The final hidden vector is:

$$h^t = o^t * \tanh(S^t) \quad (2.9)$$

For the details and variants of LSTM, we refer to (Greff et al., 2017).

2.2.2 GRU

GRU simplifies LSTM by combining the forget and input gates into an update gate:

$$z^t = \sigma(W_z h^{t-1} + U_z x^t + b_z) \quad (2.10)$$

$$h^t = (1 - z_t)h^{t-1} + z_t \hat{h}^t \quad (2.11)$$

where

$$\hat{h}^t = \tanh(W_r h^{t-1} + U_r x^t + b) \quad (2.12)$$

and where r^t is reset gate:

$$r_t = \sigma(W_r h^{t-1} + U_r x^t + b_r) \quad (2.13)$$

The details of GRU are described in (Cho et al., 2014).

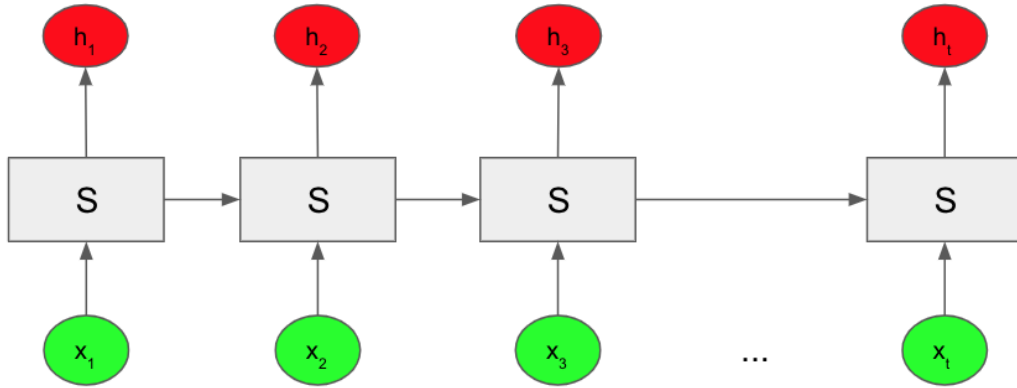


Figure 2.2: Recurrent Neural Network

2.2.3 Seq2seq

Sequence-to-sequence (Seq2seq) architecture is used to handle a pair of input and output text, e.g., text summarization or headline generation (Rush et al., 2015). It typically consists of an encoder and a decoder. The encoder reads into text input and encodes it into a sequence of vectors. These vectors are then passed to the decoder. The decoder then generates the output text step by step. We show a simple example in Figure 2.3. This example is for text summarization, where the encoder reads the article and the decoder then generates the summary for the article. For the details of seq2seq, we refer to (Sutskever et al., 2014; Rush et al., 2015). We will also elaborate it further in Chapter 6.

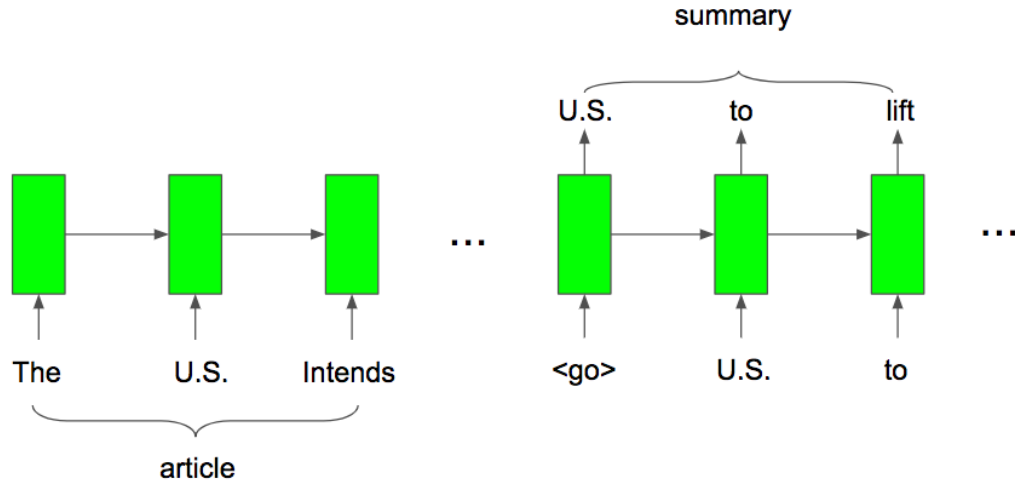


Figure 2.3: Sequence to Sequence Network

2.3 Low Supervision Scenarios

In this thesis, we consider neural NLP models under low supervision scenarios. Neural NLP models include the above mentioned CNN, RNN and seq2seq models. We define low supervision as the case when we don't have large scale training labels for the task at hand. For example, when doing sentiment analysis,

we don't have access to industry level data size such as millions of available sentiment labels that Amazon could have.

Most of prior work on low supervision scenarios uses transfer learning or domain adaptation (Glorot et al., 2011). Though there is not large scale labels for the target task at hand, there could be a large amount of data from another relevant task or domain. Rather than train the model on the target domain from scratch, people typically first pre-train a model on the relevant domain. Since the relevant domain contains enough data, the pre-trained model is assumed to be powerful. Then this model is fine-tuned on the target domain. There are many variants on pre-training and fine-tuning options (Mou et al., 2016), for example, people can pre-train word embedding on a huge unlabeled corpus, and use the pre-trained word embedding as a better initialization in the target model and fine-tune them during later training; people also pre-train a whole network on the source domain, and later only fine-tunes several top layers. For another example, in machine translation, there is some low-resource language where there are not enough labels, people pre-train the whole network on a richly annotated pair like English and French, and directly adapt this model to the low-resource pair (Wu et al., 2016b). Most recently, people pre-train a deep language model on large corpus that could capture the context information in text, and then fine-tune the language model for specific tasks. This kind of method has achieved state of the art performance across multiple NLP tasks (Howard and Ruder, 2018a; Devlin et al., 2018; Peters et al., 2018).

Chapter 3

Neural Models Augmented with Human Rationales

3.1 Chapter Overview

When human annotators label training data, they could also provide rationales that support their labels without too much extra cost (Zaidan et al., 2007). In this way, the model could not only utilize the direct training labels, but also leverage these additional rationales that can potentially augment the training. Specifically when collecting labels for long document classification, annotators can provide which sentences in the document support their annotations.

In this chapter, we present a new CNN model for text classification that jointly exploits labels on documents and their constituent sentences (Zhang et al., 2016a)¹. Specifically, we consider scenarios in which annotators explicitly mark sentences (or snippets) that support their overall document categorization, i.e., they provide *rationales*. Our model exploits such supervision via a hierarchical approach in which each document is represented by a linear combination of the vector representations of its component sentences. We propose a sentence-level convolutional model that estimates the probability that a given sentence is a rationale, and we then scale the contribution of each sentence to the aggregate document representation in proportion to these estimates. Experiments on five classification datasets that have document labels and associated rationales demonstrate that our approach consistently outperforms strong baselines. Moreover, our model naturally provides explanations for its predictions.

Specific contributions of this chapter as follows. (1) This is the first work to incorporate rationales into neural models for text classification. (2) Empirically, we show that the proposed model uniformly outperforms relevant baseline approaches

¹Ye Zhang, Iain Marshall and Byron Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. I proposed the idea, wrote the code, and wrote the paper.

across five datasets, including previously proposed models that capitalize on rationales (Zaidan et al., 2007; Marshall et al., 2016) and multiple baseline CNN variants, including a CNN equipped with an attention mechanism. We also report state-of-the-art results on the important task of automatically assessing the risks of bias in the studies described in full-text biomedical articles (Marshall et al., 2016). (3) Our model naturally provides *explanations* for its predictions, providing interpretability.

3.2 Prior Work

3.2.1 Exploiting *rationales*

In long documents the importance of sentences varies; some are more central than others. Prior work has investigated methods to measure the relative importance sentences (Ko et al., 2002; Murata et al., 2000). In this work we adopt a particular view of sentence importance in the context of document classification. In particular, we assume that documents comprise sentences that directly support their categorization. We call such sentences *rationales*.

The notion of rationales was first introduced by Zaidan et al. (2007). To harness these for classification, they proposed modifying the Support Vector Machine (SVM) objective function to encode a preference for parameter values that result in instances containing manually annotated rationales being more confidently classified than ‘pseudo’-instances from which these rationales had been stripped. This approach dramatically outperformed baseline SVM variants that do not exploit such rationales. Yessenalina et al. (2010) later developed an approach to *generate* rationales.

Another line of related work concerns models that capitalize on *dual supervision*, i.e., labels on individual *features*. This work has largely involved inserting constraints into the learning process that favor parameter values that align with *a priori* feature-label affinities or rankings (Druck et al., 2008; Mann and McCallum, 2010; Small et al., 2011; Settles, 2011). We do not discuss this line of work further here, as our focus is on exploiting provided rationales, rather than individual

labeled features.

3.3 Rationale Augmented Neural Models

We now move to the main contribution of this work: a rationale-augmented CNN for text classification. We first introduce a simple variant of the CNN (Section 2.1) that models document structure (Section 3.3.1) and then introduce a means of incorporating rationale-level supervision into this model (Section 3.3.2). In Section 3.3.3 we discuss connections to attention mechanisms and describe a baseline equipped with one, inspired by Yang et al. (2016).

3.3.1 Modeling Document Structure

Recall that rationales are snippets of text marked as having supported document-level categorizations. We aim to develop a model that can exploit these annotations during training to improve classification. Here we achieve this by developing a hierarchical model that estimates the probabilities of individual sentences being rationales and uses these estimates to inform the document level classification.

As a first step, we extend the CNN model above to explicitly account for document structure. Specifically, we apply a CNN to each individual sentence in a document to obtain sentence vectors independently. We then sum the respective sentence vectors to create a document vector.² As before, we add a softmax layer on top of the document-level vector to perform classification. We perform regularization by applying dropout both on the individual sentence vectors and the final document vector. We will refer to this model as *Doc-CNN*. Doc-CNN forms the basis for our novel approach, described below.

3.3.2 RA-CNN

In this section we present the Rationale-Augmented CNN (*RA-CNN*). Briefly, RA-CNN induces a document-level vector representation by taking a *weighted* sum

²We also experimented with taking the average of sentence vectors, but summing performed better in informal testing.

of its constituent sentence vectors. Each sentence weight is set to reflect the estimated probability that it is a rationale in support of the most likely class. We provide a schematic of this model in Figure 3.1.

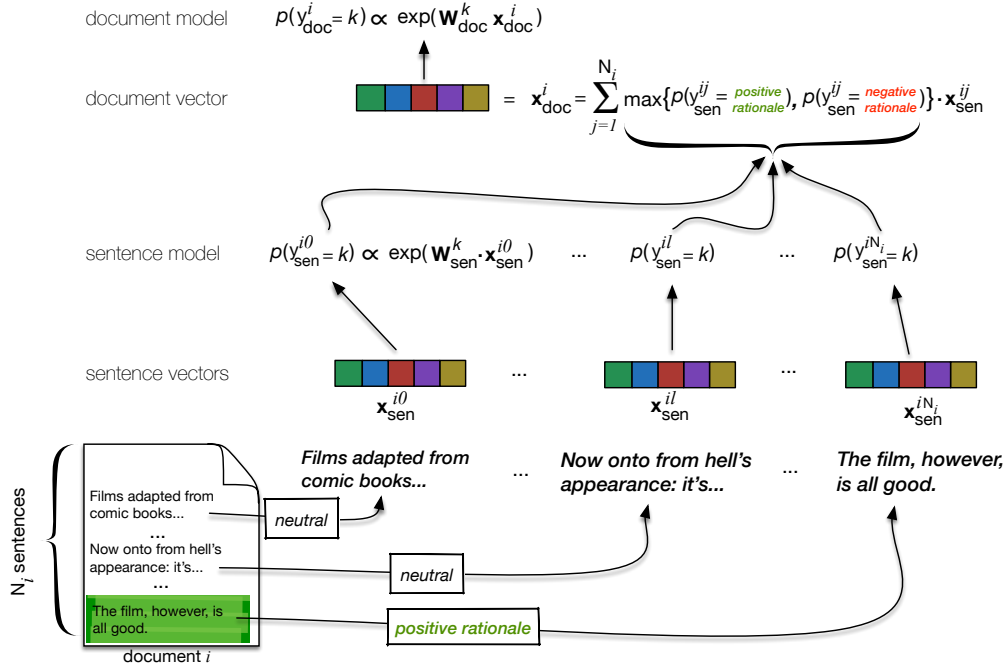


Figure 3.1: A schematic of our proposed Rationale-Augmented Convolution Neural Network (RA-CNN). The sentences comprising a text are passed through a sentence model that outputs probabilities encoding the likelihood that sentences are neutral or a (positive or negative) rationale. Sentences likely to be rationales are given higher weights in the global document vector, which is the input to the document model.

RA-CNN capitalizes on both sentence- and document-level supervision. There are thus two steps in the training phase: *sentence level* training and *document level* training. For the former, we apply a CNN to each sentence j in document i to obtain sentence vectors \mathbf{x}_{sen}^{ij} . We then add a softmax layer parametrized by \mathbf{W}_{sen} ; this takes as input sentence vectors. We fit this model to maximize the probabilities of

the observed rationales:

$$p(y_{\text{sen}}^{ij} = k; \mathbf{E}, \mathbf{C}, \mathbf{W}_{\text{sen}}) = \frac{\exp(\mathbf{W}_{\text{sen}}^{(k)T} \mathbf{x}_{\text{sen}}^{ij})}{\sum_{k=1}^{K_{\text{sen}}} \exp(\mathbf{W}_{\text{sen}}^{(k)T} \mathbf{x}_{\text{sen}}^{ij})} \quad (3.1)$$

Where y_{sen}^{ij} denotes the rationale label for sentence j in document i , K_{sen} denotes the number of possible classes for sentences, \mathbf{E} denotes the word embedding matrix, \mathbf{C} denotes the convolution layer parameters, and \mathbf{W}_{sen} is a matrix of weights (comprising one weight vector per sentence class).

In our setting, each sentence has three possible labels ($K_{\text{sen}} = 3$). When a rationale sentence appears in a positive document,³ it is a *positive rationale*; when a rationale sentence appears in a negative document, it is a *negative rationale*. All other sentences belong to a third, neutral class: these are *non-rationales*. We also experimented with having only two sentence classes: *rationales* and *non-rationales*, but this did not perform as well as explicitly maintaining separate classes for rationales of different polarities.

We train an estimator using the provided rationale annotations, optimizing over $\{\mathbf{E}, \mathbf{C}, \mathbf{W}_{\text{sen}}\}$ to minimize the categorical cross-entropy of sentence labels. Once trained, this sub-model can provide conditional probability estimates regarding whether a given sentence is a positive or a negative rationale, which we will denote by p_{pos} and p_{neg} , respectively.

We next train the document-level classification model. The inputs to this are vector representations of documents, induced by summing over constituent sentence vectors, as in Doc-CNN. However, in the RA-CNN model this is a weighted sum. Specifically, weights are set to the estimated probabilities that corresponding sentences are rationales in the most likely direction. More precisely:

$$\mathbf{x}_{\text{doc}}^i = \sum_{j=1}^{N_i} \mathbf{x}_{\text{sen}}^{ij} \cdot \max\{p_{\text{pos}}^{ij}, p_{\text{neg}}^{ij}\} \quad (3.2)$$

Where N_i is the number of sentences in the i th document. The intuition is that sen-

³ All of the document classification tasks we consider here are binary, although extension of our model to multi-class scenarios is straight-forward.

tences likely to be rationales will have greater influence on the resultant document vector representation, while the contribution of neutral sentences (which are less relevant to the classification task) will be minimized.

The final classification is performed by a softmax layer parameterized by \mathbf{W}_{doc} ; the inputs to this layer are the document vectors. The \mathbf{W}_{doc} parameters are trained using the document-level labels, y_{doc}^i :

$$p(y_{\text{doc}}^i = k; \mathbf{E}, \mathbf{C}, \mathbf{W}_{\text{doc}}) = \frac{\exp(\mathbf{W}_{\text{doc}}^{(k)T} \mathbf{x}_{\text{doc}}^i)}{\sum_{k=1}^{K_{\text{doc}}} \exp(\mathbf{W}_{\text{doc}}^{(k)T} \mathbf{x}_{\text{doc}}^i)} \quad (3.3)$$

where K_{doc} is the cardinality of the document label set. We optimize over parameters to minimize cross-entropy loss (w.r.t. the document labels).

We note that the sentence- and document-level models share word embeddings \mathbf{E} and convolution layer parameters \mathbf{C} , but the document-level model has its own softmax parameters \mathbf{W}_{doc} . When training the document-level model, \mathbf{E} , \mathbf{C} and \mathbf{W}_{doc} are fit, but we hold \mathbf{W}_{sen} fixed.

The above two-step strategy can be equivalently described as follows. We first estimate \mathbf{E} , \mathbf{C} and \mathbf{W}_{sen} , which parameterize our model for identifying rationales in documents. We then move to fitting our document classification model. For this we initialize the word embedding and convolution parameters to the \mathbf{E} and \mathbf{C} estimates from the preceding step. We then directly minimize the document level classification objective, tuning \mathbf{E} and \mathbf{C} and simultaneously fitting \mathbf{W}_{doc} .

Note that this sequential training strategy differs from the *alternating* training approach commonly used in multi-task learning (Collobert and Weston, 2008). We found that the latter approach does not work well here, leading us to instead adopt the cascade-like feature learning approach (Collobert and Weston, 2008) just described.

One nice property of our model is that it naturally provides explanations for its predictions: the model identifies rationales and then categorizes documents informed by these. Thus if the model classifies a test instance as *positive*, then by construction the sentences associated with the highest p_{pos}^{ij} estimates are those that the model relied on most in coming to this disposition. These sentences can of

course be output in conjunction with the prediction. We provide concrete examples of this in Section 3.6.2.

3.3.3 Rationales as ‘Supervised Attention’

One may view RA-CNN as a supervised variant of a model equipped with an *attention mechanism* (Bahdanau et al., 2014). On this view, it is apparent that rather than capitalizing on rationales directly, we could attempt to let the model learn which sentences are important, using *only* the document labels. We therefore construct an additional baseline that does just this, thereby allowing us to assess the impact of learning directly from rationale-level supervision.

Following the recent work of (Yang et al., 2016), we first posit for each sentence vector a hidden representation $\mathbf{u}_{\text{sen}}^{ij}$. We then define a sentence-level *context vector* \mathbf{u}_s , which we multiply with each $\mathbf{u}_{\text{sen}}^{ij}$ to induce a weight α_{ij} . Finally, the document vector is taken as a weighted sum over sentence vectors, where weights reflect α ’s. We have:

$$\mathbf{u}_{\text{sen}}^{ij} = \tanh(\mathbf{W}_s \mathbf{x}_{\text{sen}}^{ij} + b_s) \quad (3.4)$$

$$\alpha_{ij} = \frac{\exp(\mathbf{u}_s^T \mathbf{u}_{\text{sen}}^{ij})}{\sum_j^{N_i} \exp(\mathbf{u}_s^T \mathbf{u}_{\text{sen}}^{ij})} \quad (3.5)$$

$$\mathbf{x}_{\text{doc}}^i = \sum_j^{N_i} \alpha_{ij} \mathbf{x}_{\text{sen}}^{ij} \quad (3.6)$$

where $\mathbf{x}_{\text{doc}}^i$ again denotes the document vector fed into a softmax layer, and \mathbf{W}_s , \mathbf{u}_s and b_s are learned during training. We will refer to this attention-based method as *AT-CNN*.

3.4 Datasets

We used five text classification datasets to evaluate our approach in total. Four of these are biomedical text classification datasets (3.4.1) and the last is a

collection of movie reviews (3.4.2). These datasets share the property of having recorded rationales associated with each document categorization. We summarize attributes of all datasets used in this work in Table 3.1.

3.4.1 Risk of Bias (RoB) Datasets

We used a collection *Risk of Bias* (RoB) text classification datasets, described at length elsewhere (Marshall et al., 2016). Briefly, the task concerns assessing the reliability of the evidence presented in full-text biomedical journal articles that describe the conduct and results of randomized controlled trials (RCTs). This involves, e.g., assessing whether or not patients were properly blinded as to whether they were receiving an active treatment or a comparator (such as a placebo). If such blinding is not done correctly, it compromises the study by introducing statistical bias into the treatment efficacy estimate(s) derived from the trial.

A formal system for making bias assessments is codified by the Cochrane Risk of Bias Tool (Higgins et al., 2011). This tool defines multiple *domains*; the risk of bias may be assessed in each of these. We consider four domains here. (1) Random sequence generation (RSG): were patients were assigned to treatments in a truly random fashion? (2) Allocation concealment (AC): were group assignments revealed to the person assigning patients to groups (so that she may have knowingly or unknowingly) influenced these assignments? (3) Blinding of Participants and Personnel (BPP): were all trial participants and individuals involved in running the trial blinded as to who was receiving which treatment? (4) Blinding of outcome assessment (BOA): were the parties who measured the outcome(s) of interest blinded to the intervention group assignments? These assessments are somewhat subjective. To increase transparency, researchers performing RoB assessment therefore record rationales (sentences from articles) supporting their assessments.

3.4.2 Movie Review Dataset

We also ran experiments on a movie review (MR) dataset with accompanying rationales. Pang and Lee (2004) developed and published the original version of

	<i>N</i>	<i>#sen</i>	<i>#token</i>	<i>#rat</i>
RSG	8399	300	9.92	0.31
AC	11512	297	9.87	0.15
BPP	7997	296	9.95	0.21
BOA	2706	309	9.92	0.2
MR	1800	32.6	21.2	8.0

Table 3.1: Dataset characteristics. *N* is the number of instances, *#sen* is the average sentence count, *#token* is the average token per-sentence count and *#rat* is the average number of rationales per document.

this dataset, which comprises 1000 positive and 1000 negative movie reviews from the Internet Movie Database (IMDB).⁴ Zaidan et al. (2007) then augmented this dataset by adding rationales corresponding to the binary classifications for 1800 documents, leaving the remaining 200 for testing. Because 200 documents is a modest test sample size, we ran 9-fold cross validation on the 1800 annotated documents (each fold comprising 200 documents). The rationales, as originally marked in this dataset, were sub-sentential snippets; for the purposes of our model, we considered the entire sentences containing the marked snippets as rationales.

3.5 Experimental Setup

3.5.1 Baselines

We compare against several baselines to assess the advantages of directly incorporating rationale-level supervision into the proposed CNN architecture. We describe these below.

SVMs. We evaluated a few variants of linear Support Vector Machines (SVMs). These rely on sparse representations of text. We consider variants that exploit uni- and bi-grams; we refer to these as *uni-SVM* and *bi-SVM*, respectively. We also re-implemented the rationale augmented SVM (*RA-SVM*) proposed by Zaidan et al. (2007), described in Section 4.2.

For the RoB dataset, we also compare to a recently proposed multi-task

⁴<http://www.imdb.com/>

SVM (MT-SVM) model developed specifically for these RoB datasets (Marshall et al., 2015, 2016). This model exploits the intuition that the risks of bias across the domains codified in the aforementioned Cochrane RoB tool will likely be correlated. That is, if we know that a study exhibits a high risk of bias for one domain, then it seems reasonable to assume it is at an elevated risk for the remaining domains. Furthermore, Marshall et al. (2016) include rationale-level supervision by first training a (multi-task) *sentence-level* model to identify sentences likely to support RoB assessments in the respective domains. Special features extracted from these predicted rationales are then activated in the *document-level* model, informing the final classification. This model is the state-of-the-art on this task.

CNNs. We compare against several baseline CNN variants to demonstrate the advantages of our approach. We emphasize that our focus in this work is **not** to explore how to induce generally ‘better’ document vector representations – this question has been addressed at length elsewhere, e.g., (Le and Mikolov, 2014; Jozefowicz et al., 2015; Tang et al., 2015; Yang et al., 2016).

Rather, the main contribution here is an augmentation of CNNs for text classification to capitalize on rationale-level supervision, thus improving performance and enhancing interpretability. This informed our choice of baseline CNN variants: standard CNN (Kim, 2014), Doc-CNN (described above) and AT-CNN (also described above) that capitalizes on an (unsupervised) attention mechanism at the sentence level, described in Section 3.3.3.⁵

3.5.2 Implementation/Hyper-Parameter Details

Sentence splitting. To split the documents from all datasets into sentences for consumption by our Doc-CNN and RA-CNN models, we used the Natural Language Toolkit (NLTK)⁶ sentence splitter.

SVM-based models. We kept the 50,000 most frequently occurring features in each

⁵We also experimented briefly with LSTM and GRU (Gated Recurrent Unit) models, but found that simple CNN performed better than these. Moreover, CNNs are relatively robust and less sensitive to hyper-parameter selection.

⁶<http://www.nltk.org/api/nltk.tokenize.html>

Method	RSG	AC	BPP	BOA
<i>Uni-SVM</i>	72.16	72.81	72.80	65.85
<i>Bi-SVM</i>	74.82	73.62	75.13	67.29
<i>RA-SVM</i>	72.54	74.11	75.15	66.29
<i>MT-SVM</i>	76.15	74.03	76.33	67.50
<i>CNN</i>	72.50 (72.22, 72.65)	72.16 (71.49, 72.93)	75.03 (74.16, 75.44)	63.76 (63.12, 64.15)
<i>Doc-CNN</i>	72.60 (72.43, 72.90)	72.92 (72.19, 73.48)	74.24 (74.03, 74.38)	63.64 (63.23, 64.37)
<i>AT-CNN</i>	74.14 (73.40, 74.58)	73.66 (73.12, 73.92)	74.29 (74.09, 74.74)	63.34 (63.21, 63.49)
<i>RA-CNN</i>	77.42 (77.33, 77.59)	76.14 (75.89, 76.29)	76.47 (76.15, 76.75)	69.67 (69.33, 69.93)
<i>Human</i>	85.00	80.00	78.10	83.20

Table 3.2: Accuracies on the four RoB datasets. Uni-SVM: unigram SVM, Bi-SVM: Bigram SVM, RA-SVM: Rationale-augmented SVM (Zaidan et al., 2007), MT-SVM: a multi-task SVM model specifically designed for the RoB task, which also exploits the available sentence supervision (Marshall et al., 2016). We also report an estimate of human-level performance, as calculated using subsets of the data for each domain that were assessed by two experts (one was arbitrarily assumed to be correct). We report these numbers for reference; they are not directly comparable to the cross-fold estimates reported for the models.

dataset. For estimation we used SGD. We tuned the C hyper-parameter using nested development sets. For the RA-SVM, we additionally tuned the μ and $C_{contrast}$ parameters, as per (Zaidan et al., 2007).

CNN-based models. For all models and datasets we initialized word embeddings to pre-trained vectors fit via Word2Vec. For the movie reviews dataset these were 300-dimensional and trained on Google News.⁷ For the RoB datasets, these were 200-dimensional and trained on biomedical texts in PubMed/PubMed Central (Pyysalo et al., 2013).⁸

Training proceeded as follows. We first extracted all sentences from all documents in the training data. The distribution of sentence types is highly imbalanced (nearly all are neutral). Therefore, we downsampled sentences before each epoch, so that sentence classes were equally represented. After training on sentence-level supervision, we moved to document-level model fitting. For this we initialized embedding and convolution layer parameters to the estimates from the preceding sentence-level training step (though these were further tuned to optimize the document-level objective).

⁷<https://code.google.com/archive/p/word2vec/>

⁸<http://bio.nlplab.org/>

For RA-CNN, we tuned the dropout rate (range: 0-.9) applied at the sentence vector level on each training fold (using a subset of the training data as a validation set) during the document level training phase. Anecdotally, we found this has a greater effect than the other model hyperparameters, which we thus set after a small informal process of experimentation on a subset of the data. Specifically, we fixed the dropout rate at the document level to 0.5, and we used 3 different filter heights: 3, 4 and 5, following (Zhang and Wallace, 2015b). For each filter height, we used 100 feature maps for the baseline CNN, and 20 for all the other CNN variants.

For parameter estimation we used ADADELTA (Zeiler, 2012a), mini-batches of size 50, and an early stopping strategy (using a validation set).

3.6 Results and Discussion

3.6.1 Quantitative Results

For all CNN models, we replicated experiments 5 times, where each replication constituted 5-fold and 9-fold CV respectively the RoB and the movies datasets, respectively. We report the mean and observed ranges in accuracy across these 5 replications for these models, because attributes of the model (notably, dropout) and the estimation procedure render model fitting stochastic (Zhang and Wallace, 2015b). We do not report ranges for SVM-based models because the variance inherent in the estimation procedure is much lower for these simpler, linear models.

Results on the RoB datasets and the movies dataset are shown in Tables 3.2 and Table 3.3, respectively. RA-CNN consistently outperforms all of the baseline models, across all five datasets. We also observe that CNN/Doc-CNN do not necessarily improve over the results achieved by SVM-based models, which prove to be strong baselines for longer document classification. This differs from previous comparisons in the context of classifying shorter texts. In particular, in previous work (Zhang and Wallace, 2015b) we observed that CNN outperforms SVM uniformly on sentence classification tasks (the average sentence-length in these datasets was about 10). In contrast, in the datasets we consider in the present paper, documents often comprise hundreds of sentences, each in turn containing multiple words. We

Method	Accuracy
<i>Uni-SVM</i>	86.44
<i>Bi-SVM</i>	86.94
<i>RA-SVM</i>	88.89
<i>CNN</i>	85.59 (85.27, 86.17)
<i>Doc-CNN</i>	87.14 (86.70, 87.60)
<i>AT-CNN</i>	86.69 (86.28, 87.17)
<i>RA-CNN</i>	90.43 (90.11, 91.00)

Table 3.3: Accuracies on the movie review dataset.

believe that it is in these cases that explicitly modeling which sentences are most important will result in the greatest performance gains, and this aligns with our empirical results.

Another observation is that AT-CNN does often improve performance over vanilla variants of CNN (i.e., without attention), especially on the RoB datasets, probably because these comprise longer documents. However, as one might expect, RA-CNN clearly outperforms AT-CNN by exploiting rationale-level supervision directly. And by exploiting rationale information directly, RA-CNN is able to consistently perform better than baseline CNN and SVM model variants. Indeed, we find that RA-CNN outperformed MT-SVM on all of the RoB datasets, and this was accomplished without exploiting cross-domain correlations (i.e., without multi-task learning).

Although RA-CNN is consistently better than the other CNNs with same amount of document labels, it also incurs extra labeling efforts on rationales. Zaidan et al. (2007) claims that labeling both the documents and their rationales might cause at least twice as much time as labeling only documents. A more fair comparison between RA-CNN and baselines should be with same amount of human labeling efforts (with baseline CNNs having more annotated document labels). This is left to future exploration.

3.6.2 Qualitative Results: Illustrative Rationales

In addition to realizing superior classification performance, RA-CNN also provides *explainable* categorizations. The model can provide the highest scoring

rationales (ranked by $\max\{p_{\text{pos}}, p_{\text{neg}}\}$) for any given target instance, which in turn – by construction – are those that most influenced the final document classification.

For example, a sample positive rationale supporting a correct designation of a study as being at low risk of bias with respect to blinding of outcomes assessment reads simply *The study was performed double blind*. An example rationale extracted for a study (correctly) deemed at high risk of bias, meanwhile, reads *as the present study is retrospective, there is a risk that the woman did not properly recall how and what they experienced*

Turning to the movie reviews dataset, an example rationale extracted from a glowing review of ‘Goodfellas’ (correctly classified as positive) reads *this cinematic gem deserves its rightful place among the best films of 1990s*. While a rationale extracted from an unfavorable review of ‘The English Patient’ asserts that *the only redeeming qualities about this film are the fine acting of Fiennes and Dafoe and the beautiful desert cinematography*.

In each of these cases, the extracted rationales directly support the respective classifications. This provides direct, meaningful insight into the automated classifications, an important benefit for neural models, which are often seen as opaque.

3.7 Chapter Summary

We developed a new model (RA-CNN) for text classification that extends the CNN architecture to directly exploit *rationales* when available. We showed that this model outperforms several strong, relevant baselines across five datasets, including vanilla and hierarchical CNN variants, and a CNN model equipped with an attention mechanism. Moreover, RA-CNN automatically provides explanations for classifications made at test time, thus providing interpretability.

Moving forward, we plan to explore additional mechanisms for exploiting supervision at lower levels in neural architectures. Furthermore, we believe an alternative approach may be a hybrid of the AT-CNN and RA-CNN models, wherein an auxiliary loss might be incurred when the attention mechanism output disagrees with the available direct supervision on sentences.

Chapter 4

Neural Models Augmented with Ontologies

4.1 Chapter Overview

When the training size is not big enough, neural models often suffer from overfitting. One of the most effective approaches to overcome this issue is introducing some bias into the model, e.g., regularization. Most often, these bias are totally random without any prior domain knowledge. However, in practice this often means ignoring existing external linguistic resources, e.g., WordNet or domain specific ontologies such as the Unified Medical Language System (UMLS). In this chapter, we propose a general, novel method for exploiting such resources via *weight sharing* (Zhang et al., 2017b)¹. Prior work on weight sharing in neural networks has considered it largely as a means of model compression. In contrast, we treat weight sharing as a flexible mechanism for incorporating prior knowledge into neural models.

We propose exploiting the *feature-hashing* trick, originally proposed as a means of neural network compression (Chen et al., 2015). Here we instead view the partial parameter sharing induced by feature hashing as a flexible mechanism for tying together network node weights that we believe to be similar *a priori*. In effect, this acts as a regularizer that constrains the model to learn weights that agree with the domain knowledge codified in external resources like ontologies.

More specifically, as external resources we use Brown clusters (Brown et al., 1992), WordNet (Miller, 1995) and the Unified Medical Language System (UMLS) (Bodenreider, 2004). From these we derive groups of words with similar meaning. We then use feature hashing to share a subset of weights between the embeddings of words that belong to the same semantic group(s). This forces the model to re-

¹Ye Zhang, Matthew Lease and Byron Wallace. 2017. Exploiting Domain Knowledge via Grouped Weight Sharing with Application to Text Categorization. In *Proceedings of Association for Computational Linguistics*. I proposed the idea, implemented the idea and wrote the paper.

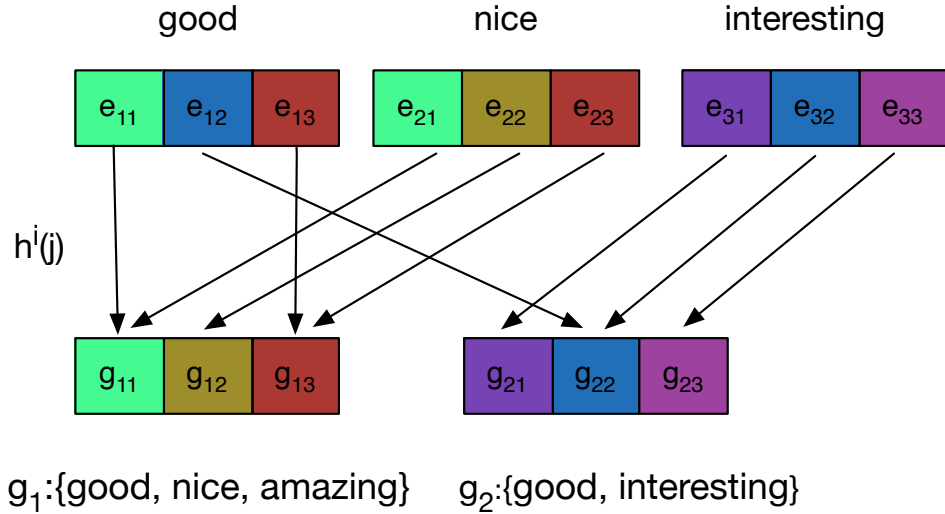


Figure 4.1: An example of grouped partial weight sharing. Here there are two groups. We stochastically select embedding weights to be shared between words belonging to the same group(s).

spect prior domain knowledge, insofar as words similar under a given ontology are compelled to have similar embeddings.

Our contribution is a novel, simple and flexible method for injecting domain knowledge into neural models via stochastic weight sharing. Results on seven diverse classification tasks (three sentiment and four biomedical) show that our method consistently improves performance over (1) baselines that fail to capitalize on domain knowledge, and (2) an approach that uses *retrofitting* (?) as a preprocessing step to encode domain knowledge prior to training.

4.2 Prior Work

Exploiting Linguistic Resources. A potential drawback to learning from scratch in end-to-end neural models is a failure to capitalize on existing knowledge sources. There have been efforts to exploit such resources specifically to induce better word vectors (Yu and Dredze, 2014; Faruqui et al., 2014; Yu et al., 2016; Xu et al., 2014). But these models do not attempt to exploit external resources jointly during training

for a particular downstream task (which uses word embeddings as inputs), as we do here.

Past work on sparse linear models has shown the potential of exploiting linguistic knowledge in statistical NLP models. For example, Yogatama and Smith (2014) used external resources to inform structured, grouped regularization of log-linear text classification models, yielding improvements over standard regularization approaches. Elsewhere, Doshi-Velez et al. (2015) proposed a variant of LDA that exploits *a priori* known tree-structured relations between tokens (e.g., derived from the UMLS) in topic modeling.

Weight-sharing in NNs. Recent work has considered stochastically sharing weights in neural models. Notably, Chen et al. (2015). proposed randomly sharing weights in neural networks. Elsewhere, Han et al. (2015) proposed quantized weight sharing as an intermediate step in their deep compression model. In these works, the primary motivation was *model compression*, whereas here we view the hashing trick as a mechanism to encode domain knowledge.

4.3 Weight Sharing in Embedding Space

We incorporate similarity relations codified in existing resources (here derived from Brown clusters, SentiWordNet and the UMLS) as prior knowledge in a Convolutional Neural Network (CNN).² To achieve this we construct a shared embedding matrix such that words known *a priori* to be similar are constrained to share some fraction of embedding weights.

Concretely, suppose we have N groups of words derived from an external resource. Note that one could derive such groups in several ways; e.g., using the synsets in SentiWordNet. We denote groups by $\{g_1, g_2, \dots, g_N\}$. Each group is associated with an embedding \mathbf{g}_{g_i} , which we initialize by averaging the pre-trained embeddings of each word in the group.

To exploit both grouped and independent word weights, we adopt a two-

²The idea of sharing weights to reflect known similarity is general and could be applied with other neural architectures.

channel CNN model (Zhang et al., 2016b). The embedding matrix of the first channel is initialized with pre-trained word vectors. We denote this input by $\mathbf{E}^p \in \mathbb{R}^{V \times d}$ (V is the vocabulary size and d the dimension of the word embeddings). The second channel input matrix is initialized with our proposed weight-sharing embedding $\mathbf{E}^s \in \mathbb{R}^{V \times d}$. \mathbf{E}^s is initialized by drawing from both \mathbf{E}^p and the external resource following the process we describe below.

Given an input text sequence of length l , we construct sequence embedding representations $\mathbf{W}^p \in \mathbb{R}^{l \times d}$ and $\mathbf{W}^s \in \mathbb{R}^{l \times d}$ using the corresponding embedding matrices. We then apply independent sets of linear convolution filters on these two matrices. Each filter will generate a feature map vector $\mathbf{v} \in \mathbb{R}^{l-h+1}$ (h is the filter height). We perform 1-max pooling over each \mathbf{v} , extracting one scalar per feature map. Finally, we concatenate scalars from all of the feature maps (from both channels) into a feature vector which is fed to a softmax function to predict the label (Figure 4.2).

We initialize \mathbf{E}^s as follows. Each row $\mathbf{e}_i \in \mathbb{R}^d$ of \mathbf{E}_s is the embedding of word i . Words may belong to one or more groups. A mapping function $G(i)$ retrieves the groups that word i belongs to, i.e., $G(i)$ returns a subset of $\{g_1, g_2, \dots, g_N\}$, which we denote by $\{g_1^{(i)}, g_2^{(i)} \dots g_K^{(i)}\}$, where K is the number of groups that contain word i . To initialize \mathbf{E}^s , for each dimension j of each word embedding \mathbf{e}_i , we use a hash function h^i to map (hash) the index j to one of the K group IDs: $h^i : \mathbb{N} \rightarrow \{g_1^{(i)}, g_2^{(i)} \dots g_K^{(i)}\}$. Following (Weinberger et al., 2009; Shi et al., 2009), we use a second hash function b to remove bias induced by hashing. This is a signing function, i.e., it maps (i, j) tuples to $\{+1, -1\}$.³ We then set $\mathbf{e}_{i,j}$ to the product of $\mathbf{g}_{h^i(j),j}$ and $b(i, j)$. h and b are both approximately uniform hash functions. **Algorithm 1** provides the full initialization procedure.

For illustration, consider Figure 4.1. Here g_1 contains three words: *good*, *nice* and *amazing*, while g_2 has two words: *good* and *interesting*. The group embeddings \mathbf{g}_{g_1} , \mathbf{g}_{g_2} are initialized as averages over the pre-trained embeddings of the words they comprise. Here, embedding parameters $\mathbf{e}_{1,1}$ and $\mathbf{e}_{2,1}$ are both mapped to $\mathbf{g}_{g_1,1}$, and thus share this value. Similarly, $\mathbf{e}_{1,3}$ and $\mathbf{e}_{2,3}$ will share value at $\mathbf{g}_{g_1,3}$.

³Empirically, we found that using this signing function does not affect performance.

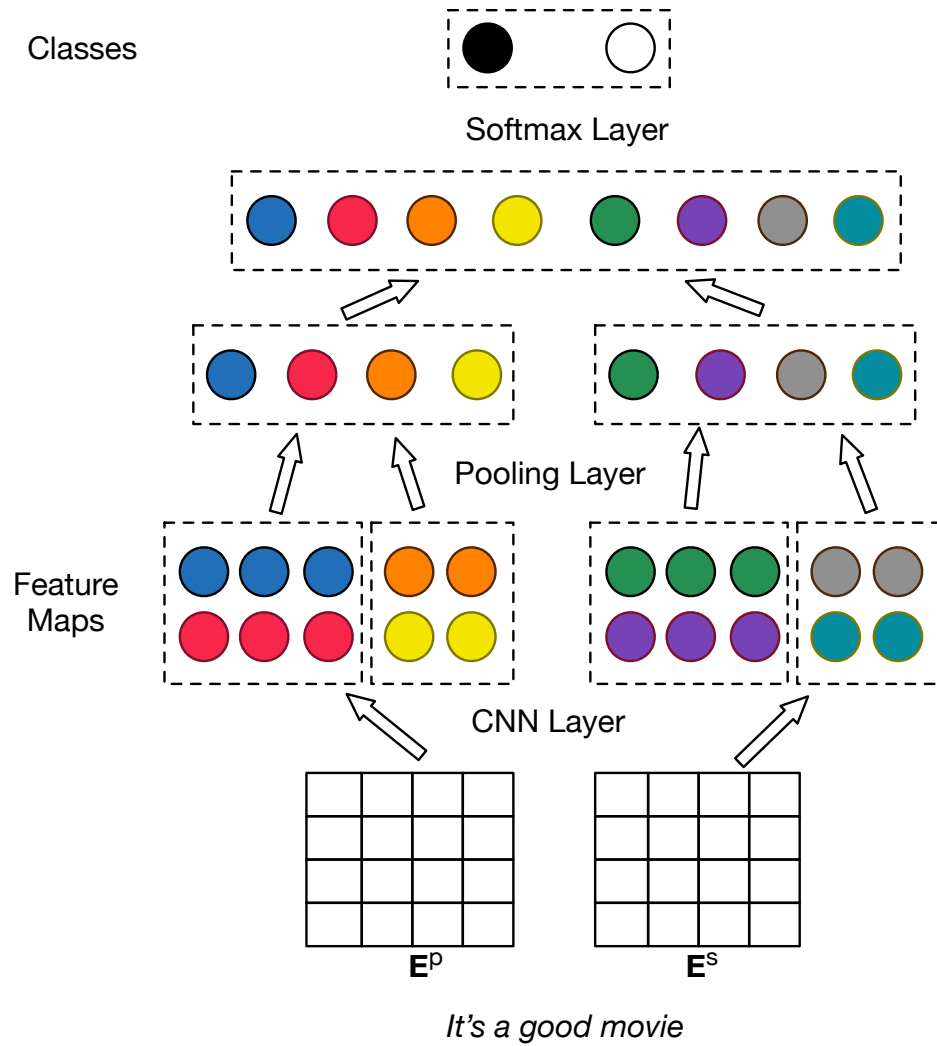


Figure 4.2: Proposed two-channel model. The first channel input is a standard pre-trained embedding matrix. The second channel receives a partially shared embedding matrix constructed using external linguistic resources.

Algorithm 1 Initialization of \mathbf{E}^s

```
1: for  $i$  in  $\{1, \dots, V\}$  do
2:    $\{g_1^{(i)}, g_2^{(i)}, \dots, g_K^{(i)}\} := G(i)$ .
3:   for  $j \in \{1, \dots, d\}$  do
4:      $\mathbf{e}_{i,j} := \mathbf{g}_{h^i(j),j} \cdot b(i, j)$ 
5:   end for
6: end for
```

We have elided the second hash function b from this figure for simplicity.

During training, we update \mathbf{E}^p as usual using back-propagation (Rumelhart et al., 1986). We update \mathbf{E}^s and group embeddings \mathbf{g} in a manner similar to (Chen et al., 2015). In the forward propagation before each training step (mini-batch), we derive the value of $\mathbf{e}_{i,j}$ from \mathbf{g} :

$$\mathbf{e}_{i,j} := \mathbf{g}_{h^i(j),j} * b(i, j) \quad (4.1)$$

We use this newly updated $\mathbf{e}_{i,j}$ to perform forward propagation in the CNN.

During backward propagation, we first compute the gradient of \mathbf{E}^s , and then we use this to derive the gradient w.r.t \mathbf{g}_s . To do this, for each dimension j in \mathbf{g}_{g_k} , we aggregate the gradients w.r.t \mathbf{E}^s whose elements are mapped to this dimension:

$$\nabla \mathbf{g}_{g_k,j} := \sum_{(i,j)} \nabla \mathbf{E}_{i,j}^s \cdot \delta_{h^i(j)=g_k} \cdot b(i, j) \quad (4.2)$$

where $\delta_{h^i(j)=g_k} = 1$ when $h^i(j) = g_k$, and 0 otherwise. Each training step involves executing Equations 4.1 and 4.2. Once the shared gradient is calculated, gradient descent proceeds as usual. We update all parameters aside from the shared weights in the standard way.

The number of parameters in our approach scales linearly with the number of channels. But the gradients can actually be back-propagated in a distributed way for each channel, since the convolutional and embedding layers are independent across these. Thus training time scales approximately linearly with the number of parameters in one channel (if the gradient is back-propagated in a distributed way).

	<i>total #instances</i>	<i>vocabulary size</i>	<i>#positive instances</i>	<i>#negative instances</i>
MR	10662	18765	5331	5331
CR	3773	5340	2406	1367
MPQA	10604	6246	3311	7293
AN	5653	5554	653	5000
CL	8288	3684	768	7520
ST	3464	2965	173	3291
PB	4749	3086	243	4506

Table 4.1: Corpora statistics.

4.4 Experimental Setup

4.4.1 Datasets

We use three sentiment datasets: a movie review (**MR**) dataset (Pang and Lee, 2005a)⁴; a customer review (**CR**) dataset (Hu and Liu, 2004)⁵; and an opinion dataset (**MPQA**) (Wiebe et al., 2005)⁶.

We also use four biomedical datasets, which concern *systematic reviews*. The task here is to classify published articles describing clinical trials as *relevant* or *not* to a well-specified clinical question. Articles deemed relevant are included in the corresponding review, which is a synthesis of all pertinent evidence (Wallace et al., 2010). We use data from reviews that concerned: clopidogrel (CL) for cardiovascular conditions (Dahabreh et al., 2013); biomarkers for assessing iron deficiency in anemia (AN) experienced by patients with kidney disease (Chung et al., 2012); statins (ST) (Cohen et al., 2006); and proton beam (PB) therapy (Terasawa et al., 2009).

4.4.2 Implementation Details and Baselines

We use SentiWordNet (Baccianella et al., 2010)⁷ for the sentiment tasks. SentiWordNet assigns to each synset of wordnet three sentiment scores: positivity,

⁴www.cs.cornell.edu/people/pabo/movie-review-data/

⁵www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

⁶mpqa.cs.pitt.edu/corpora/mpqa_corpus/

⁷sentiwordnet.isti.cnr.it

negativity and objectivity, constrained to sum to 1. We keep only the synsets with positivity or negativity scores greater than 0, i.e., we remove synsets deemed objective. The synsets in SentiWordNet constitute our groups. We also use the Brown clustering algorithm⁸ on the three sentiment datasets. We generate 1000 clusters and treat each as a group.

For the biomedical datasets, we use the Medical Subject Headings (MeSH) terms⁹ attached to each abstract to classify them. Each MeSH term has a tree number indicating the path from the root in the UMLS. For example, ‘Alagille Syndrome’ has tree number ‘C06.552.150.125’; periods denote tree splits, numbers are nodes. We induce groups comprising MeSH terms that share the same first three parent nodes, e.g., all terms with ‘C06.552.150’ as their tree number prefix constitute one group.

We compare our approach to several baselines. All use pre-trained embeddings to initialize E^p , but we explore several approaches to exploiting E^s : (1) randomly initialize E^s ; (2) initialize E^s to reflect the group embedding g , but do not share weights during the training process, i.e., do not constrain their weights to be equal when we perform back-propagation; (3) use linguistic resources to *retro-fit* (Faruqui et al., 2014) the pre-trained embeddings, and use these to initialize E^s . For *retro-fitting*, we first construct a graph derived from SentiWordNet. Then we run belief-propagation on the graph to encourage linked words to have similar vectors. This is a pre-processing step only; we do not impose weight sharing constraints during training.

For the sentiment datasets we use three filter heights (3,4,5) for each of the two CNN channels. For the biomedical datasets, we use only one filter height (1), because the inputs are unstructured MeSH terms.¹⁰ In both cases we use 100 filters of each unique height. For the sentiment datasets, we use Google word2vec (Mikolov et al., 2013a)¹¹ to initialize E^p . For the biomedical datasets, we use

⁸github.com/percyliang/brown-cluster

⁹www.nlm.nih.gov/bsd/disted/meshtutorial/

¹⁰For this work we are ignoring title and abstract texts.

¹¹code.google.com/archive/p/word2vec/

word2vec trained on biomedical texts (Moen and Ananiadou, 2013)¹² to initialize E^p . For parameter estimation, we use Adadelta (Zeiler, 2012b). Because the biomedical datasets are imbalanced, we use downsampling (Zhang et al., 2016a; Zhang and Wallace, 2015a) to effectively train on balanced subsets of the data.

We developed our approach using the MR sentiment dataset, tuning our approach to constructing groups from the available resources – experiments on other sentiment datasets were run after we finalized the model and hyperparameters. Similarly, we used the anemia (AN) review as a development set for the biomedical tasks, especially w.r.t. constructing groups from MeSH terms using UMLS.

4.5 Results

We replicate each experiment five times (each is a 10-fold cross validation), and report the mean (min, max) across these replications. Results on the sentiment and biomedical corpora are presented in Tables 4.2 and 4.3, respectively.¹³ These exploit different external resources to induce the word groupings that in turn inform weight sharing. We report AUC for the biomedical datasets because these are highly imbalanced (see Table 4.1).

Our method improves performance compared to all relevant baselines (including an approach that also exploits external knowledge via retrofitting) in six of seven cases. Informing weight *initialization* using external resources improves performance independently, but additional gains are realized by also enforcing sharing during training.

We note that our aim here is not necessarily to achieve state-of-art results on any given dataset, but rather to evaluate the proposed method for incorporating external linguistic resources into neural models via weight sharing. We have therefore compared to baselines that enable us to assess this.

¹²bio.nlplab.org/

¹³Sentiment task results are not directly comparable to prior work due to different preprocessing steps.

<i>Method</i>	<i>MR</i>	<i>CR</i>	<i>MPQA</i>
p only	81.02 (80.84,81.24)	84.34 (84.21,84.53)	89.41 (89.22,89.58)
p + r	81.25 (81.19,81.32)	84.33 (84.24,84.38)	89.63 (89.58,89.71)
p + retro	81.35 (81.23,81.51)	84.16 (84.09,84.28)	89.61 (89.48,89.77)
p + S (no sharing)	81.39 (81.32,81.43)	84.13 (84.06,84.21)	89.71 (89.67,89.75)
p + B (no sharing)	81.50 (81.29,81.63)	84.60 (84.53,84.66)	89.57 (89.52,89.61)
p + S (sharing)	81.69 (81.60,81.78)	84.34 (84.24,84.43)	89.84 (89.74,90.13)
p + B (sharing)	81.83 (81.80,81.87)	84.68 (84.64,84.72)	89.97 (89.74,90.13)

Table 4.2: Accuracy mean (min, max) on sentiment datasets. ‘p’: channel initialized with the pre-trained embeddings E^p . ‘r’: channel randomly initialized. ‘retro’: initialized with retrofitted embeddings. ‘S/B (no sharing)’: channel initialized with E^s (using SentiWordNet or Brown clusters), but weights are not shared during training. ‘S/B (sharing)’: proposed weight-sharing method.

<i>Method</i>	<i>AN</i>	<i>CL</i>	<i>ST</i>	<i>PB</i>
p only	86.63 (86.57,86.67)	88.73 (88.51,89.00)	67.15 (66.00, 67.91)	90.11 (89.46, 91.03)
p + r	85.67 (85.46,85.95)	88.87 (88.56,89.03)	67.72 (67.65,67.86)	90.12 (89.87,90.47)
p + retro	86.46 (86.32,86.65)	89.27 (88.89,90.01)	67.78 (67.56,68.00)	90.07 (89.92,90.20)
p + U	86.60 (86.32,87.01)	88.93 (88.67,89.13)	67.78 (67.71,67.85)	90.23 (89.84,90.47)
p + U(s)	87.15 (87.00,87.29)	89.29 (89.09,89.51)	67.73 (67.58,67.88)	90.99 (90.46,91.59)

Table 4.3: AUC mean (min, max) on the biomedical datasets. Abbreviations are as in Table 4.2, except here the external resource is the UMLS MeSH ontology (‘U’). ‘U(s)’ is the proposed weight sharing method utilizing ULMS.

4.6 Chapter Summary

We have proposed a novel method for incorporating prior semantic knowledge into neural models via stochastic weight sharing. We have showed it generally improves text classification performance vs. model variants which do not exploit external resources and vs. an approach based on retrofitting prior to training. In future work, we will investigate generalizing our approach beyond classification, and to inform weight sharing using other varieties and sources of linguistic knowledge.

Chapter 5

Neural Models Augmented with Multiple Pre-trained Embeddings

5.1 Chapter Overview

One of the most commonly used method that handles low-supervision situations is transfer learning (Pan and Yang, 2010). Specifically in NLP, researchers often pre-train word embeddings on arbitrarily large corpora (Mikolov et al., 2013b). Then we can use these pre-trained embeddings to initialize the word embeddings for our tasks and fine-tune them to the task at hand. This has been shown to work better than train word embeddings from scratch (Zhang and Wallace, 2015a). In this chapter, we introduce a novel, simple CNN architecture – *multi-group norm constraint CNN (MGNC-CNN)* – that capitalizes on multiple sets of word embeddings for sentence classification (Zhang et al., 2016b)¹. MGNC-CNN extracts features from input embedding sets independently and then joins these at the penultimate layer in the network to form a final feature vector. We then adopt a group regularization strategy that differentially penalizes weights associated with the subcomponents generated from the respective embedding sets. This model is much simpler than comparable alternative architectures and requires substantially less training time. Furthermore, it is flexible in that it does not require input word embeddings to be of the same dimensionality. We show that MGNC-CNN consistently outperforms baseline models.

¹Ye Zhang, Stephen Roller and Byron Wallace. 2016. MGNC-CNN: A Simple Approach to Exploiting Multiple Word Embeddings for Sentence Classification. In *Proceedings of North American Chapter of the Association for Computational Linguistics*. I proposed the idea, implemented the idea and wrote the paper.

5.2 Prior Work

We introduce neural models for text classification in Chapter 2. An important consideration for neural models is the specification of the word embeddings. Several options exist. For example, Kalchbrenner et al. (2014) initialize word vectors to random low-dimensional vectors to be fit during training, while Johnson and Zhang (2014) use fixed, one-hot encodings for each word. By contrast, Kim (2014) initializes word vectors to those estimated via the word2vec model trained on 100 billion words of Google News (Mikolov et al., 2013b); these are then updated during training. Initializing embeddings to pre-trained word vectors is intuitively appealing because it allows transfer of learned distributional semantics. This has allowed a relatively simple CNN architecture to achieve remarkably strong results.

Many pre-trained word embeddings are now readily available on the web, induced using different models, corpora, and processing steps. Different embeddings may encode different aspects of language (Padó and Lapata, 2007; Erk and Padó, 2008; Levy and Goldberg, 2014): those based on bag-of-words (BoW) statistics tend to capture associations (*doctor* and *hospital*), while embeddings based on dependency-parses encode similarity in terms of use (*doctor* and *surgeon*). It is natural to consider how these embeddings might be *combined* to improve NLP models in general and CNNs in particular.

Prior work has considered combining latent representations of words that capture syntactic and semantic properties (Van de Cruys et al., 2011), and inducing multi-modal embeddings (Bruni et al., 2012) for general NLP tasks. And recently, Luo et al. (2014) proposed a framework that combines multiple word embeddings to measure text similarity, however their focus was not on classification.

More similar to our work, Yin and Schütze (2015) proposed *MVCNN* for sentence classification. This CNN-based architecture accepts multiple word embeddings as inputs. These are then treated as separate ‘channels’, analogous to RGB channels in images. Filters consider all channels simultaneously. MVCNN achieved state-of-the-art performance on multiple sentence classification tasks. However, this model has practical drawbacks. (i) MVCNN requires that input word em-

beddings have the same dimensionality. Thus to incorporate a second set of word vectors trained on a corpus (or using a model) of interest, one needs to either find embeddings that happen to have a set number of dimensions or to estimate embeddings from scratch. (ii) The model is complex, both in terms of implementation and run-time. Indeed, this model requires pre-training and mutual-learning and requires days of training time, whereas the simple architecture we propose requires on the order of an hour (and is easy to implement). We directly compare our model with MVCNN in the experiment.

5.3 Method

MG-CNN. Assuming we have m word embeddings with corresponding dimensions d_1, d_2, \dots, d_m , we can simply treat each word embedding independently. In this case, the input to the CNN comprises multiple sentence matrices $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_m$, where each $\mathbf{A}_l \in \mathbb{R}^{s \times d_l}$ may have its own width d_l . We then apply different groups of filters $\{\mathbf{w}_1\}, \{\mathbf{w}_2\}, \dots, \{\mathbf{w}_m\}$ independently to each \mathbf{A}_l , where $\{\mathbf{w}_l\}$ denotes the set of filters for \mathbf{A}_l . As in basic CNN, $\{\mathbf{w}_l\}$ may have multiple filter sizes, and multiple filters of each size may be introduced. At the classification layer we then obtain a feature vector \mathbf{o}_l for each embedding set, and we can simply concatenate these together to form the final feature vector \mathbf{o} to feed into the softmax function, where $\mathbf{o} = \mathbf{o}_1 \oplus \mathbf{o}_2 \dots \oplus \mathbf{o}_m$. This representation contains feature vectors generated from all sets of embeddings under consideration. We call this method *multiple group CNN* (MG-CNN). Here groups refer to the features generated from different embeddings. Note that this differs from ‘multi-channel’ models because at the convolution layer we use different filters on each word embedding matrix independently, whereas in a standard multi-channel approach each filter would consider all channels simultaneously and generate a scalar from all channels at each local region. As above, we impose a max l2 norm constraint on the final feature vector \mathbf{o} for regularization. Figure 5.1 illustrates this approach.

MGNC-CNN. We propose an augmentation of MG-CNN, *Multi-Group Norm Constraint CNN* (MGNC-CNN), which differs in its regularization strategy. Specif-

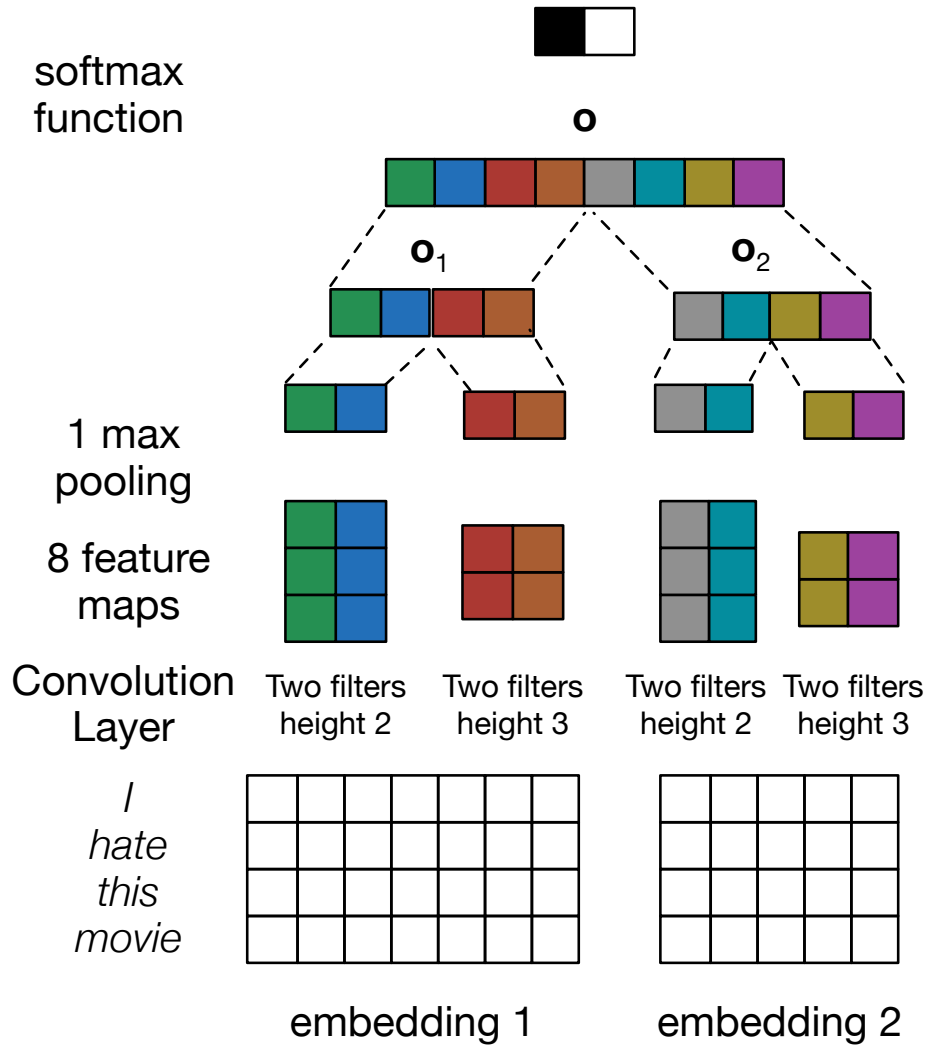


Figure 5.1: Illustration of MG-CNN and MGNC-CNN. The filters applied to the respective embeddings are completely independent. MG-CNN applies a max norm constraint to \mathbf{o} , while MGNC-CNN applies max norm constraints on \mathbf{o}_1 and \mathbf{o}_2 independently (group regularization). Note that one may easily extend the approach to handle more than two embeddings at once.

ically, in this variant we impose *grouped* regularization constraints, independently regularizing subcomponents \mathbf{o}_l derived from the respective embeddings, i.e., we

<i>Model</i>	<i>Subj</i>	<i>SST-1</i>	<i>SST-2</i>	<i>TREC</i>	<i>Irony</i>
CNN(w2v)	93.14 (92.92,93.39)	46.99 (46.11,48.28)	87.03 (86.16,88.08)	93.32 (92.40,94.60)	67.15 (66.53,68.11)
CNN(Glv)	93.41(93.20,93.51)	46.58 (46.11,47.06)	87.36 (87.20,87.64)	93.36 (93.30,93.60)	67.84 (67.29,68.38)
CNN(Syn)	93.24(93.01,93.45)	45.48(44.67,46.24)	86.04 (85.28,86.77)	94.68 (94.00,95.00)	67.93 (67.30,68.38)
MVCNN (Yin and Schütze, 2015)	93.9	49.6	89.4	-	-
C-CNN(w2v+Glv)	93.72 (93.68,93.76)	47.02(46.24,47.69)	87.42(86.88,87.81)	93.80 (93.40,94.20)	67.70 (66.97,68.35)
C-CNN(w2v+Syn)	93.48 (93.43,93.52)	46.91(45.97,47.81)	87.17 (86.55,87.42)	94.66 (94.00,95.20)	68.08 (67.33,68.57)
C-CNN(w2v+Syn+Glv)	93.61 (93.47,93.77)	46.52 (45.02,47.47)	87.55 (86.77,88.58)	95.20 (94.80,95.60)	68.38 (67.66,69.23)
MG-CNN(w2v+Glv)	93.84 (93.66,94.35)	48.24 (47.60,49.05)	87.90 (87.48,88.30)	94.09 (93.60,94.80)	69.40 (66.35,72.30)
MG-CNN(w2v+Syn)	93.78 (93.62,93.98)	48.48(47.78,49.19)	87.47(87.10,87.70)	94.87 (94.00,95.60)	68.28 (66.44,69.97)
MG-CNN(w2v+Syn+Glv)	94.11 (94.04,94.17)	48.01 (47.65,48.37)	87.63(87.04,88.36)	94.68 (93.80,95.40)	69.19(67.06,72.30)
MGNC-CNN(w2v+Glv)	93.93 (93.79,94.14)	48.53 (47.92,49.37)	88.35(87.86,88.74)	94.40 (94.00,94.80)	69.15 (67.25,71.70)
MGNC-CNN(w2v+Syn)	93.95 (93.75,94.21)	48.51 (47.60,49.41)	87.88(87.64,88.19)	95.12 (94.60,95.60)	69.35 (67.40,70.86)
MGNC-CNN(w2v+Syn+Glv)	94.09 (93.98,94.18)	48.65 (46.92,49.19)	88.30 (87.83,88.65)	95.52 (94.60,96.60)	71.53 (69.74,73.06)

Table 5.1: Results mean (min, max) achieved with each method. w2v:word2vec. Glv:GloVe. Syn: Syntactic embedding. Note that we experiment with using two and three sets of embeddings jointly, e.g., w2v+Syn+Glv indicates that we use all three of these.

<i>Model</i>	<i>Subj</i>	<i>SST-1</i>	<i>SST-2</i>	<i>TREC</i>	<i>Irony</i>
CNN(w2v)	9	81	81	9	243
CNN(Glv)	3	9	1	9	81
CNN(Syn)	3	81	9	81	1
C-CNN(w2v+Glv)	9	9	3	3	1
C-CNN(w2v+Syn)	3	81	9	9	1
C-CNN(w2v+Syn+Glv)	9	9	1	81	81
MG-CNN(w2v+Glv)	3	9	3	81	9
MG-CNN(w2v+Syn)	9	81	3	81	3
MG-CNN(w2v+Syn+Glv)	9	1	9	243	9
MGNC-CNN(w2v+Glv)	(9,3)	(81,9)	(1,1)	(9,81)	(243,243)
MGNC-CNN(w2v+Syn)	(3,3)	(81,81)	(81,9)	(81,81)	(81,3)
MGNC-CNN(w2v+Syn+Glv)	(81,81,81)	(81,81,1)	(9,9,9)	(1,81,81)	(243,243,3)

Table 5.2: Best λ^2 value on the validation set for each method w2v:word2vec. Glv:GloVe. Syn: Syntactic embedding.

impose separate max norm constraints λ_l for each \mathbf{o}_l (where l again indexes embedding sets); these λ_l hyper-parameters are to be tuned on a validation set. Intuitively, this method aims to better capitalize on features derived from word embeddings that capture discriminative properties of text for the task at hand by penalizing larger weight estimates for features derived from less discriminative embeddings.

5.4 Experiments

5.4.1 Datasets

Stanford Sentiment Treebank Stanford Sentiment Treebank (SST) (Socher et al., 2013). This concerns predicting movie review sentiment. Two datasets are derived from this corpus: (1) **SST-1**, containing five classes: *very negative*, *negative*, *neutral*, *positive*, and *very positive*. (2) **SST-2**, which has only two classes: *negative* and *positive*. For both, we remove phrases of length less than 4 from the training set.² **Subj** (Pang and Lee, 2004). The aim here is to classify sentences as either *subjective* or *objective*. This comprises 5000 instances of each. **TREC** (Li and Roth, 2002). A question classification dataset containing six classes: *abbreviation*, *entity*, *description*, *human*, *location* and *numeric*. There are 5500 training and 500 test instances. **Irony** (Wallace et al., 2014a). This dataset contains 16,006 sentences from reddit labeled as ironic (or not). The dataset is imbalanced (relatively few sentences are ironic). Thus before training, we under-sampled negative instances to make classes sizes equal. Note that for this dataset we report the Area Under Curve (AUC), rather than accuracy, because it is imbalanced.

5.4.2 Pre-trained Word Embeddings

We consider three sets of word embeddings for our experiments: (i) **word2vec**³ is trained on 100 billion tokens of Google News dataset; (ii) **GloVe** (Pennington et al., 2014)⁴ is trained on aggregated global word-word co-occurrence statistics from Common Crawl (840B tokens); and (iii) **syntactic** word embedding trained on dependency-parsed corpora. These three embedding sets happen to all be 300-dimensional, but our model could accommodate arbitrary and variable sizes.

We pre-trained our own syntactic embeddings following (Levy and Goldberg, 2014). We parsed the ukWaC corpus (Baroni et al., 2009) using the Stanford Dependency Parser v3.5.2 with Stanford Dependencies (Chen and Manning, 2014)

²As in (Kim, 2014).

³<https://code.google.com/p/word2vec/>

⁴<http://nlp.stanford.edu/projects/glove/>

and extracted (word, relation+context) pairs from parse trees. We “collapsed” nodes with prepositions and notated inverse relations separately, e.g., “dog barks” emits two tuples: $(barks, nsubj_dog)$ and $(dog, nsubj^{-1}_barks)$. We filter words and contexts that appear fewer than 100 times, resulting in $\sim 173k$ words and 1M contexts. We trained 300d vectors using word2vec⁵ with default parameters.

5.4.3 Setup

We compared our proposed approaches to MVCNN (Yin and Schütze, 2015) and also a standard CNN that exploits a single set of word embeddings (Kim, 2014). We also compared to a baseline of simply concatenating embeddings for each word to form long vector inputs. We refer to this as Concatenation-CNN *C-CNN*. For all multiple embedding approaches (C-CNN, MG-CNN and MGNC-CNN), we explored two combined sets of embedding: word2vec+Glove, and word2vec+syntactic, and one three sets of embedding: word2vec+Glove+syntactic. For all models, we tuned the l2 norm constraint λ over the range $\{\frac{1}{3}, 1, 3, 9, 81, 243\}$ on a validation set. For instantiations of MGNC-CNN in which we exploited two embeddings, we tuned both λ_1 , and λ_2 ; where we used three embedding sets, we tuned λ_1 , λ_2 and λ_3 .

We used standard train/test splits for those datasets that had them. Otherwise, we performed 10-fold cross validation, creating nested development sets with which to tune hyperparameters. For all experiments we used filters sizes of 3, 4 and 5 and we created 100 feature maps for each filter size. We applied 1 max-pooling and dropout (rate: 0.5) at the classification layer. For training we used back-propagation in mini-batches and used AdaDelta as the stochastic gradient descent (SGD) update rule, and set mini-batch size as 50. In this work, we treat word embeddings as part of the parameters of the model, and update them as well during training. In all our experiments, we only tuned the max norm constraint(s), fixing all other hyperparameters.

⁵<https://bitbucket.org/yoavgo/word2vecf/>

5.4.4 Results and Discussion

We repeated each experiment 10 times and report the mean and ranges across these. This replication is important because training is stochastic and thus introduces variance in performance (Zhang and Wallace, 2015a). Results are shown in Table 5.1, and the corresponding best norm constraint value is shown in Table 5.2. We also show results on *Subj*, *SST-1* and *SST-2* achieved by the more complex model of (Yin and Schütze, 2015) for comparison; this represents the state-of-the-art on the three datasets other than TREC.

We can see that **MGNC-CNN and MG-CNN always outperform baseline methods (including C-CNN), and MGNC-CNN is usually better than MG-CNN**. And on the *Subj* dataset, MG-CNN actually achieves slightly better results than (Yin and Schütze, 2015), with far less complexity and required training time (MGNC-CNN performs comparably, although no better, here). On the TREC dataset, the best-ever accuracy we are aware of is 96.0% (Mou et al., 2015), which falls within the range of the result of our MGNC-CNN model with three word embeddings. On the *irony* dataset, our model with three embeddings achieves 4% improvement (in terms of AUC) compared to the baseline model.

On *SST-1* and *SST-2*, our model performs slightly worse than (Yin and Schütze, 2015). However, we again note that their performance is achieved using a much more complex model which involves pre-training and mutual-learning steps. This model takes days to train, whereas our model requires on the order of an hour.

We note that the method proposed by Astudillo *et al.* 2015 is able to accommodate multiple embedding sets with different dimensions by projecting the original word embeddings into a lower-dimensional space. However, this work requires training the optimal projection matrix on labeled data first, which again incurs large overhead.

Of course, our model also has its own limitations: in MGNC-CNN, we need to tune the norm constraint hyperparameter for all the word embeddings. As the number of word embedding increases, this will increase the running time. However, this tuning procedure is embarrassingly parallel.

We mainly care about the performance improvement from the single word embedding to multiple word embedding models under exactly the same preprocessing steps. We emphasize this because we found that different preprocessing methods influence the performance, especially on SST-1 and SST-2 datasets. In previous work, authors often don't consider this factor when comparing their methods with baseline models, this is, when they compare performance of different methods, they don't remove the variance of preprocessing for each method. We also found that for all the methods, replacing a word embedding with a random vector would make the performance worse, which proves the advantages of pre-trained embeddings.

5.5 Chapter Summary

We have proposed MGNC-CNN: a simple, flexible CNN architecture for sentence classification that can exploit multiple, variable sized word embeddings. We demonstrated that this consistently achieves better results than a baseline architecture that exploits only a single set of word embeddings, and also a naive concatenation approach to capitalizing on multiple embeddings. Furthermore, our results are comparable to those achieved with a recently proposed model (Yin and Schütze, 2015) that is much more complex. However, our simple model is easy to implement and requires an order of magnitude less training time. Furthermore, our model is much more flexible than previous approaches, because it can accommodate variable-size word embeddings.

Chapter 6

Domain Adaptation for Neural Text Generation

6.1 Chapter Overview

In the last chapter, we describe utilizing multiple pre-trained embedding as a way of transfer learning for low-supervision scenarios. Although pre-training word embedding is very powerful given a large pre-training corpus, a more straightforward transfer learning method is domain adaptation (Glorot et al., 2011), in which the model is transferred between one same task. In the training data, there could be multiple domains. At test time, the test instance might come from one of the domains in the training data, but might be also from a domain that is never seen in the training data. We call the general approach that takes into consideration these domain information as domain adaptation. Specifically in this chapter, we consider domain adaption for abstractive text summarization (Zhang et al., 2018)¹. In the rest of this chapter, we use ‘domain’ and ‘style’ exchangeably since we assume text from different domains are written with different styles.

Supervised training of abstractive text summarization models results in learning conditional probabilities over language sequences based on the supervised training signal. When the training signal contains a variety of writing styles, such models may end up learning an ‘average’ style that is directly influenced by the training data make-up and cannot be controlled by the needs of an application. We describe a family of model architectures capable of capturing both generic language characteristics via shared model parameters, as well as particular style characteristics via private model parameters. Such models are able to generate language according to a specific learned style, while still taking advantage of their power to model generic language phenomena. Furthermore, we describe an extension that uses a mixture

¹Ye Zhang, Nan Ding and Radu Soricut. 2018. SHAPED: Shared-Private Encoder-Decoder for Text Style Adaptation. In *Proceedings of North American Chapter of the Association for Computational Linguistics*. I proposed part of the idea, wrote the code and wrote the paper.

of output distributions from all learned styles to perform on-the-fly style adaptation based on the textual input alone. Experimentally, we find that the proposed models consistently outperform models that encapsulate single-style or average-style language generation capabilities.

Encoder-decoder models have recently pushed forward the state-of-the-art performance on a variety of language generation tasks, including machine translation (Bahdanau et al., 2014; Wu et al., 2016b; Vaswani et al., 2017), text summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017), dialog systems (Li et al., 2016b; Asghar et al., 2017), and image captioning (Xu et al., 2015; Ranzato et al., 2015; Liu et al., 2017). This framework consists of an encoder that reads the input data and encodes it as a sequence of vectors, which is in turn used by a decoder to generate another sequence of vectors used to produce output symbols step by step.

The prevalent approach to training such a model is to update all the model parameters using all the examples in the training data (over multiple epochs). This is a reasonable approach, under the assumption that we are modeling a single underlying distribution in the data. However, in many applications and for many natural language datasets, there exist multiple underlying distributions, characterizing a variety of language styles. For instance, the widely-used Gigaword dataset (Graff and Cieri, 2003) consists of a collection of articles written by various publishers (The New York Times, Agence France Presse, Xinhua News, etc.), each with its own style characteristics. Training a model’s parameters on all the training examples results in an averaging effect across style characteristics, which may lower the quality of the outputs; additionally, this averaging effect may be completely undesirable for applications that require a level of control over the output style. At the opposite end of the spectrum, one can choose to train one independent model per each underlying distribution (assuming we have the appropriate signals for identifying them at training time). This approach misses the opportunity to exploit common properties shared by these distributions (e.g., generic characteristics of a language, such as noun-adjective position), and leads to models that are under-trained due to limited data availability per distribution.

In order to address these issues, we propose a novel neural architecture called SHARED-Private Encoder-Decoder (SHAPED). This architecture has both shared encoder/decoder parameters that are updated based on all the training examples, as well as private encoder/decoder parameters that are updated using only examples from their corresponding underlying training distributions. In addition to learning different parametrization between the shared model and the private models, we jointly learn a classifier to estimate the probability of each example belonging to each of the underlying training distributions. In such a setting, the shared parameters (‘shared model’) are expected to learn characteristics shared by the entire set of training examples (i.e., language generic), whereas each private parameter set (‘private model’) learns particular characteristics (i.e., style specific) of their corresponding training distribution. At the same time, the classifier is expected to learn a probability distribution over the labels used to identify the underlying distributions present in the input data. At test time, there are two possible scenarios. In the first one, the input signal explicitly contains information about the underlying distribution (e.g., the publisher’s identity). In this case, we feed the data into the shared model and also the corresponding private model, and perform sequence generation based on a concatenation of their vector outputs; we refer to this model as the SHAPED model. In a second scenario, the information about the underlying distribution is either not available, or it refers to a distribution that was not seen during training. In this case, we feed the data into the shared model and all the private models; the output distribution of the symbols of the decoding sequence is estimated using a mixture of distributions from all the decoders, weighted according to the classifier’s estimates for that particular example; we refer to this model as the Mix-SHAPED model.

We test our models on the headline-generation task based on the aforementioned Gigaword dataset. When the publisher’s identity is presented as part of the input, we show that the SHAPED model significantly surpasses the performance of the shared encoder-decoder baseline, as well as the performance of private models (where one individual, per-publisher model is trained for each in-domain style). When the publisher’s identity is not presented as part of the input (i.e., not pre-

sented at run-time but revealed at evaluation-time for measurement purposes), we show that the Mix-SHAPED model exhibits a high level of classification accuracy based on textual inputs alone (accuracy percentage in the 80s overall, varying by individual publisher), while its generation accuracy still surpasses the performance of the baseline models. Finally, when the publisher’s identity is unknown to the model (i.e., a publisher that was not part of the training dataset), we show that the Mix-SHAPED model performance far surpasses the shared model performance, due to the ability of the Mix-SHAPED model to perform on-the-fly adaptation of output style. This feat comes from our model’s ability to perform two distinct tasks: match the incoming, previously-unseen input style to existing styles learned at training time, and use the correlations learned at training time between input and output style characteristics to generate style-appropriate token sequences.

6.2 Prior Work

Encoder-Decoder Models for Structured Output Prediction

Encoder-decoder architectures have been successfully applied to a variety of structure prediction tasks recently. Tasks for which such architectures have achieved state-of-the-art results include machine translation (Bahdanau et al., 2014; Wu et al., 2016b; Vaswani et al., 2017), automatic text summarization (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Paulus et al., 2017; Nema et al., 2017), sentence simplification (Filippova et al., 2015; Zhang and Lapata, 2017), dialog systems (Li et al., 2016b, 2017; Asghar et al., 2017), image captioning (Vinyals et al., 2015; Xu et al., 2015; Ranzato et al., 2015; Liu et al., 2017), etc. By far the most used implementation of such architectures is based on the original sequence-to-sequence model (Sutskever et al., 2014), augmented with its attention-based extension (Bahdanau et al., 2014). Although our SHAPED and Mix-SHAPED model formulations do not depend on a particular architecture implementation, we do make use of the (Bahdanau et al., 2014) model to instantiate our models.

Domain Adaptation for Neural Network Models

One general approach to domain adaptation for natural language tasks is to perform data/feature augmentation that represents inputs as both general and domain-dependent data, as originally proposed in (Daumé III, 2009), and ported to neural models in (Kim et al., 2016). For computer vision tasks, a line of work related to our approach has been proposed by Bousmalis et al. (2016) using *domain separation networks*. As a tool for studying unsupervised domain adaptation for image recognition tasks, they use CNNs for encoding an image into a feature representation, and also for reconstructing the input sample. They make use of a private encoder for each domain, and a shared encoder for both the source and the target domain. The approach we take in this paper shares this idea of model parametrization according to the domain/style, but goes further with the Mix-SHAPED model, performing on-the-fly adaptation of the model outputs. Other CNN-based domain adaptation methods for object recognition tasks are presented in (Long et al., 2016; Chopra et al., 2013; Tzeng et al., 2015; Sener et al., 2016).

For NLP tasks, Peng and Dredze (2017) take a multi-task approach to domain adaptation and sequence tagging. They use a shared encoder to represent instances from all of the domains, and use a domain projection layer to project the shared layer into a domain-specific space. They only consider the supervised domain-adaptation case, in which labeled training data exists for the target domain. Glorot et al. (2011) use auto-encoders for learning a high-level feature extraction across domains for sentiment analysis, while Zhou et al. (2016) employ auto-encoders to directly transfer the examples across different domains also for the same sentiment analysis task. Hua and Wang (2017) perform an experimental analysis on domain adaptation for neural abstractive summarization.

An important requirement of all the methods in the related work described above is that they require access to the (unlabeled) target domain data, in order to learn a domain-invariant representation across source and target domains. In contrast, our Mix-SHAPED model does not need access to a target domain or style at training time, and instead performs the adaptation on-the-fly, according to the

specifics of the input data and the correlations learned at training time between available input and output style characteristics. As such, it is a more general approach, which allows adaptation for a much larger set of target styles, under the weaker assumption that there exists one or more styles present in the training data that can act as representative underlying distributions.

6.3 Shared-Private Encoder Decoder Model for Sequence Generation

Generally speaking, a standard encoder-decoder model has two components: an encoder that takes as input a sequence of symbols $\mathbf{x} = (x_1, x_2, \dots, x_{T_x})$ and encodes them into a set of vectors $\mathbf{H} = (h_1, h_2, \dots, h_{T_x})$,

$$\mathbf{H} = f_{\text{enc}}(\mathbf{x}), \quad (6.1)$$

where f_{enc} is the computation unit in the encoder, and a decoder that generates output symbols at each time stamp t , conditioned on \mathbf{H} as well as the decoder inputs $\mathbf{y}_{1:t-1}$,

$$s_t = f_{\text{dec}}(\mathbf{y}_{1:t-1}, \mathbf{H}), \quad (6.2)$$

where f_{dec} is the computation unit in the decoder. Instantiations of this framework include the widely-used attention-based sequence-to-sequence model (Bahdanau et al., 2014), in which f_{enc} and f_{dec} are implemented by an RNN architecture using LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Chung et al., 2014) units. A more recent instantiation of this architecture is the Transformer model (Vaswani et al., 2017) built solely using self-attention layers.

6.3.1 SHAPED: Shared-private encoder-decoder

The abstract encoder-decoder model described above is usually trained over all examples in the training data. We call such a model a *shared* encoder-decoder model, because the model parameters are shared across all training and test in-

stances. Formally, the shared encoder-decoder consists of the computation units f_{enc}^s and f_{dec}^s . Given an instance \mathbf{x} , it generates a sequence of vectors $\mathbf{S}^s = (s_1^s, \dots, s_T^s)$ by:

$$\mathbf{H}^s = f_{\text{enc}}^s(\mathbf{x}), s_t^s = f_{\text{dec}}^s(\mathbf{y}_{1:t-1}, \mathbf{H}^s). \quad (6.3)$$

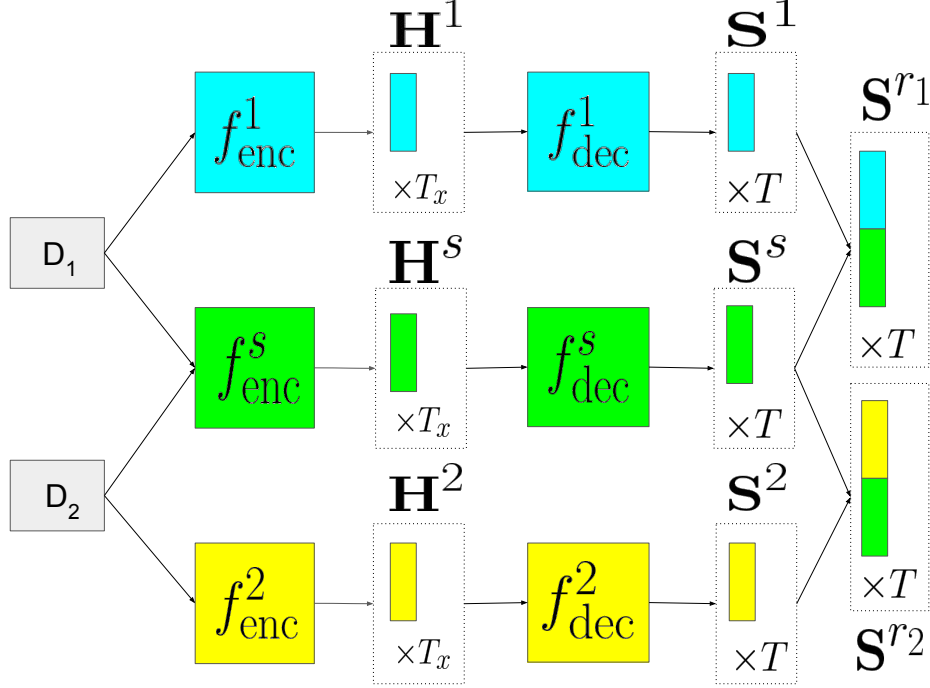


Figure 6.1: Illustration of the SHAPED model using two styles D_1 and D_2 . D_1 articles pass through the private encoder f_{enc}^1 and decoder f_{dec}^1 . D_2 articles pass through the private encoder f_{enc}^2 and decoder f_{dec}^2 . Both of them also go through the shared encoder f_{enc}^s and decoder f_{dec}^s .

The drawback of the shared encoder-decoder is that it fails to account for particular properties of each style that may be present in the data. In order to capture such particular style characteristics, a straightforward solution is to train a *private* model for each style. Assuming a style set $\mathbb{D} = \{D_1, D_2, \dots, D_{|\mathbb{D}|}\}$, such a solution implies that each style has its own private encoder computation unit and decoder computation unit. At both training and testing time, each private encoder and decoder only processes instances that belong to their own style. Given an instance

along with its style (\mathbf{x}, z) where $z \in \{1, \dots, |\mathbb{D}|\}$, the private encoder-decoder generates a sequence of vectors $\mathbf{S}^z = (s_1^z, \dots, s_T^z)$ by:

$$\mathbf{H}^z = f_{\text{enc}}^z(\mathbf{x}), s_t^z = f_{\text{dec}}^z(\mathbf{y}_{1:t-1}, \mathbf{H}^z). \quad (6.4)$$

Although the private encoder/decoder models do preserve style characteristics, they fail to take into account the common language features shared across styles. Furthermore, since each style is represented by a subset of the entire training set, such private models may end up as under-trained, due to limited number of available data examples.

In order to efficiently capture both common and unique features of data with different styles, we propose the SHARED-Private Encoder-Decoder (SHAPED) model. In the SHAPED model, each data-point goes through both the shared encoder-decoder and its corresponding private encoder-decoder. At each step of the decoder, the output from private and shared ones are concatenated to form a new vector:

$$s_t^{rz} = [s_t^z, s_t^s], \quad (6.5)$$

that contains both private features for style z and shared features induced from other styles, as illustrated in Fig 6.1. The output symbol distribution over tokens $o_t \in V$ (where V is the output vocabulary) at step t is given by:

$$p(o_t | \mathbf{x}, \mathbf{y}_{1:t-1}, z) = \text{Softmax}(g(s_t^{rz})), \quad (6.6)$$

where g is a multi-layer feed-forward network that maps s_t^{rz} to a vector of size $|V|$. Given N training examples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, z^{(1)}), \dots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)}, z^{(N)})$, the conditional probability of the output $\mathbf{y}^{(i)}$ given article $\mathbf{x}^{(i)}$ and its style $z^{(i)} \in \{1, \dots, |\mathbb{D}|\}$ is:

$$p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}, z^{(i)}) = \prod_t p(o_t = y_t^{(i)} | \mathbf{x}^{(i)}, \mathbf{y}_{1:t-1}^{(i)}, z^{(i)}). \quad (6.7)$$

At inference time, given an article \mathbf{x} with style z , we feed \mathbf{x} into $f_{\text{enc}}^z, f_{\text{dec}}^z, f_{\text{enc}}^s, f_{\text{dec}}^s$ and obtain symbol distributions at each step t using Eq:ProjSoftmax. We sample

from the distribution and obtain a symbol o_t which will be used as the estimated y_t and fed to the next steps.

6.3.2 The Mix-SHAPED Model

One limitation of the above model is that it can only handle test data containing an explicit style label from $\mathbb{D} = \{D_1, D_2, \dots, D_{|\mathbb{D}|}\}$. However, there is frequently the case that, at test time, the style label is not present as part of the input, or that the input style is not part of the modeled set \mathbb{D} .

We treat both of these cases similarly, as a case of modeling an unknown style. We first describe our treatment of such a case at run-time. We use a latent random variable $z \in \{1, \dots, |\mathbb{D}|\}$ to denote the underlying style of a given input. When generating a token at step t , the output token distribution takes the form of a mixture of SHAPED (Mix-SHAPED) model outputs:

$$p(o_t | \mathbf{x}, \mathbf{y}_{1:t-1}) = \sum_{d=1}^{|\mathbb{D}|} p(o_t | \mathbf{x}, \mathbf{y}_{1:t-1}, z = d) p(z = d | \mathbf{x}), \quad (6.8)$$

where $p(o_t | \mathbf{x}, \mathbf{y}_{1:t-1}, z = d)$ is the output symbol distribution of SHAPED decoder d , evaluated as in 6.6. Fig 6.2 contains an illustration of such a model. In this formulation, $p(\cdot | \mathbf{x})$ denotes the style conditional probability distribution given \mathbf{x} .

The joint data likelihood of target sequence \mathbf{y} and target domain label z for input sequence \mathbf{x} is:

$$p(\mathbf{y}, z | \mathbf{x}) = p(\mathbf{y} | z, \mathbf{x}) \cdot p(z | \mathbf{x}) \quad (6.9)$$

Training the Mix-SHAPED model involves minimizing a loss function that combines the negative log-likelihood of the style labels and the negative log-likelihood

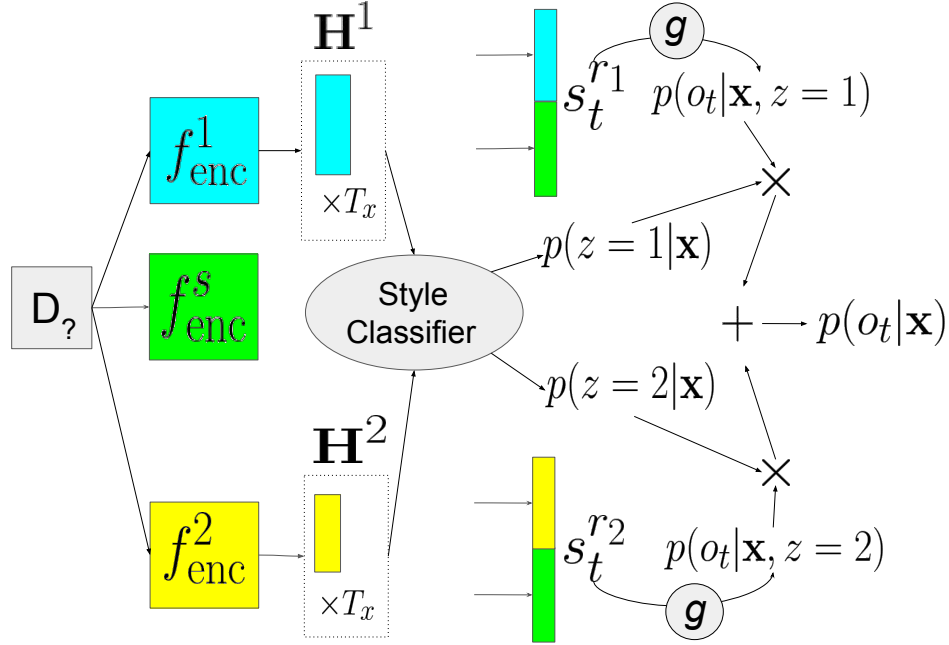


Figure 6.2: Decoding data with unknown style using a Mix-SHAPED model. The data is run through all encoders and decoders. The output of private encoders is fed into a classifier that estimates style distribution. The output symbol distribution is a mixture over all decoder outputs.

of the symbol sequences (see the model in Fig 6.3):

$$\begin{aligned}
 \text{Loss}_{\text{Mix-SHAPED}} = & - \sum_{i=1}^N \log(z^{(i)} | \mathbf{x}^{(i)}) \\
 & - \sum_{i=1}^N \log p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}, z^{(i)}).
 \end{aligned} \tag{6.10}$$

At run-time, if the style d of the input is available and $d \in \mathbb{D}$, we decode the sequence using Eq. 6.6. This also corresponds to the case $p(z = d | \mathbf{x}) = 1$ and 0 for all other styles, and reduces Eq. 6.8 to Eq. 6.6. If the style of the input is unknown (or known, but with $d' \notin \mathbb{D}$), we decode the sequence using Eq. 6.8, in

which case the mixture over SHAPED models given by $p(\cdot|\mathbf{x})$ is approximating the desired output style.

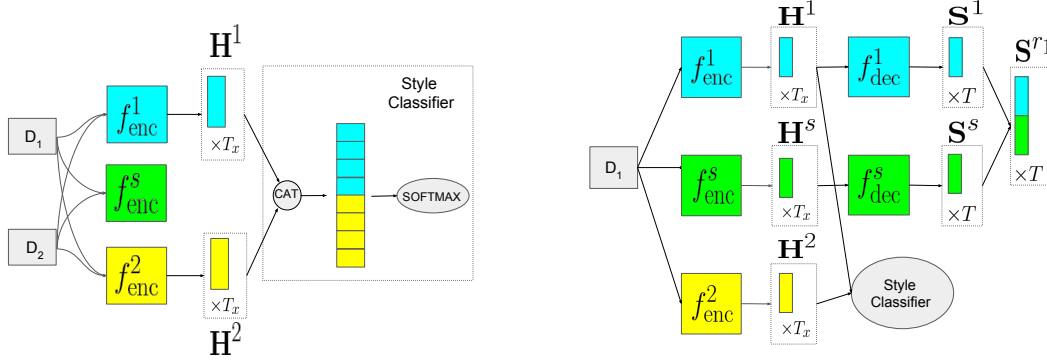


Figure 6.3: (a) Each example is fed to all private encoders f_{enc}^1, f_{enc}^2 , whose outputs are concatenated and fed to a style classifier. (b) The D_1 examples only use $f_{enc}^1, f_{dec}^1, f_{enc}^s, f_{dec}^s$ to decode texts. Private encoder-decoders of other styles are not used.

6.3.3 Model Instantiation

As an implementation of the encoder-decoder model, we use the attention-based sequence-to-sequence model from (Bahdanau et al., 2014), with an RNN architecture using GRU units (Chung et al., 2014). The input token sequences are first projected into an embedding space via an embedding matrix \mathbf{E} , resulting in a sequence of low-dimensional vectors as input representations.

6.3.4 SHAPED Instantiation

The private and shared RNN cells generate a sequence of hidden state vectors $\mathbf{H}^s = \{h_j^s\}$ and $\mathbf{H}^z = \{h_j^z\}$, $z \in \{1, \dots, |\mathbb{D}|\}$, $j \in \{1, \dots, T_x\}$. At each step in the encoder, h_j^s and h_j^z are concatenated to form a new output vector $h_j^{rz} = [h_j^z, h_j^s]$. The final state of each encoder is used as the initial state of the corresponding decoder. At time step t in the decoder, the private and shared RNN cell first generate hidden state vectors s_t^s and $\{s_t^z\}$, $z \in \{1, \dots, |\mathbb{D}|\}$, and s_t^s is concatenated with each s_t^z to form new vectors $\{s_t^{rz}\}$ ($z \in \{1, \dots, |\mathbb{D}|\}$).

We apply the attention mechanism on s_t^{rz} , using attention weights calculated as:

$$q_{tj}^{rz} = v_a \tanh(W_a h_j^{rz} + U_a s_t^{rz}), \quad (6.11)$$

which are normalized to a probability distribution:

$$\alpha_{tj}^{rz} = \frac{\exp(q_{tj}^{rz})}{\sum_{i=1}^{T_x} \exp(q_{ti}^{rz})} \quad (6.12)$$

Context vectors are computed using normalized attention weights:

$$c_t^{rz} = \sum_{j=1}^{T_x} \alpha_{tj}^{rz} h_j^{rz} \quad (6.13)$$

Given the context vector and the hidden state vectors, the symbol distribution at step t is:

$$p(o_t | \mathbf{x}, \mathbf{y}_{1:t}, z) = \text{softmax}(q[c_t^{rz}, s_t^{rz}] + b) \quad (6.14)$$

The attention weights in W_a , U_a , and v_a , as well as the embedding matrix \mathbf{E} and vocabulary V are shared by all encoders and decoders. We use eq. 6.14 to calculate the symbol loss in eq. 6.10.

6.4 Experiments

We performed a battery of quantitative experiments, designed to answer several main questions: 1) Do the proposed model improve generation performance over alternative approaches? 2) Can a style classifier built using an auxiliary loss provide a reliable estimate on text style? 3) In the case of unknown style, does the Mix-SHAPED model improve generation performance over alternative approaches? 4) To what extent do our models capture style characteristics as opposed to, say, content characteristics?

We perform our experiments using text summarization as the main task. More precisely, we train and evaluate headline generation models using the publicly-available Gigaword dataset (Graff and Cieri, 2003; Napoles et al., 2012).

The Gigaword dataset contains news articles from seven publishers: Agence France-Presse (AFP), Associated Press Worldstream (APW), Central News Agency of Taiwan (CNA), Los Angeles Times/Washington Post Newswire Service (LTW), New York Times (NYT), Xinhua News Agency (XIN), and Washington Post/Bloomberg Newswire Service (WPB). We pre-process this dataset in the same way as (Rush et al., 2015), which results in articles with average length 31.4 words and headlines with average length 8.5.

We consider the publisher identity as a proxy for style, and choose to model as in-domain styles the set $\mathbb{D} = \{\text{AFP}, \text{APW}, \text{NYT}, \text{XIN}\}$, while holding out CNA and LTW for out-of-domain style testing. This results in a training set containing the following number of (article, headline) instances: 993,584 AFP, 1,493,758 APW, 578,259 NYT, and 946,322 XIN. For the test set, we sample a total number of 10,000 in-domain examples from the original Gigawords test dataset, which include 2,886 AFP, 2,832 APW, 1,610 NYT, and 2,012 XIN. For out-of-domain testing, we randomly sample 10,000 LTW and 10,000 CNA test data examples. We remove the WPB articles due to their small number of instances.

6.4.1 Experimental Setup

We compare the following models:

- A Shared encoder-decoder model (S) trained on all styles in \mathbb{D} ;
- A suite of Private encoder-decoder models (P), each one trained on a particular style from $\mathbb{D} = \{\text{AFP}, \text{APW}, \text{NYT}, \text{XIN}\}$;²
- A SHAPED model (SP) trained on all styles in \mathbb{D} ; at test time, the style of test data is provided to the model; the article is only run through its style-specific private network and shared network (style classifier is not needed);
- A Mix-SHAPED model (M-SP) trained on all styles in \mathbb{D} ; at test time, the style of article is not provided to the model; the output is computed using the mixture model, with the estimated style probabilities from the style classifier

²We also tried to warm-start a private model using the best checkpoint of the shared model, but found that it cannot improve over the shared model.

used as weights.

When testing on the out-of-domain styles CNA/LTW, we only compare the Shared (S) model with the Mix-SHAPED (M-SP) model, as the others cannot properly handle this scenario.

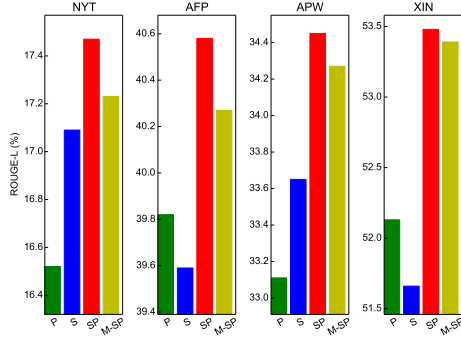
As hyper-parameters for the model instantiation, we used 500-dimension word embeddings, and a three-layer, 500-dimension GRU-cell RNN architecture; the encoder was instantiated as a bi-directional RNN. The lengths of the input and output sequences were truncated to 40 and 20 tokens, respectively. All the models were optimized using Adagrad (Duchi et al., 2011), with an initial learning rate of 0.01. The training procedure was done over mini-batches of size 128, and the updates were done asynchronously across 40 workers for 5M steps. The encoder/decoder word embedding and the output projection matrices were tied to minimize the number of parameters. To avoid the slowness from the softmax operator over large vocabulary sizes, and also mitigate the impact of out-of-vocabulary tokens, we applied a subtokenization method (Wu et al., 2016b), which invertibly transforms a native token into a sequence of subtokens from a limited vocabulary (here set to 32K).

Comparison with Previous Work

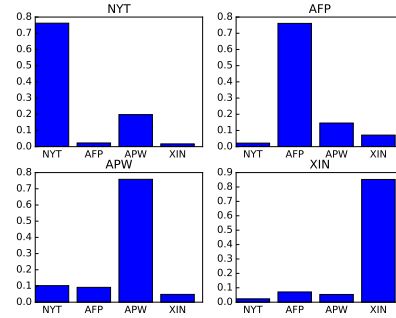
In the next section, we report our main results using the in-domain and out-of-domain (w.r.t. the selected publisher styles) test sets described above, since these test sets have a balanced publisher style frequency that allows us to measure the impact of our style-adaptation models. However, we also report here the performance of our Shared (S) baseline model (with the above hyper-parameters) on the original 2K test set used by (Rush et al., 2015). On that test set, our S model obtains 30.13 F1 ROUGE-L score, compared to 28.34 ROUGE-L obtained by the ABS+ model (Rush et al., 2015), and 30.64 ROUGE-L obtained by the words-lvt2k-1sent model (Nallapati et al., 2016). This comparison indicates that our S model is a competitive baseline, making the comparisons against the SP and M-SP models meaningful when using our in-domain and out-of-domain test sets.

6.4.2 Main Results

The Rouge scores for the in-domain testing data are reported in Table 6.1 (over the combined AFP/APW/XIN/NYT testset) and Fig. 6.4a (over individual-style test sets). The numbers indicate that the SP and M-SP models consistently outperform the S and P model, supporting the conclusion that the S model loses important characteristics due to averaging effects, while the P models miss the opportunity to efficiently exploit the training data. Additionally, the performance of SP is consistently better than M-SP in this setting, which indicates that the style label is helpful. As shown in Fig. 6.4b, the style classifier achieves around 80% accuracy overall in predicting the style under the M-SP model, with some styles (e.g., XIN) being easier to predict than others. The performance of the classifier is directly reflected in the quantitative difference between the SP and M-SP models on individual-style test sets (see Fig. 6.4a, where the XIN style has the smallest difference between the two models).



(a) Rouge-L scores on headline generation, shown separately on four in-domain styles.



(b) Average estimated probability distribution by the M-SP model over the four styles, for each in-domain target style in the test set.

Figure 6.4: Experimental results on the headline generation task, for in-domain styles.

The evaluation results for the out-of-domain scenario are reported in Table 6.2. The numbers indicate that the M-SP model significantly outperforms the

	AFP/APW/XIN/NYT Test		
	Rouge-1	Rouge-2	Rouge-L
P	39.14 \pm 0.47	19.74 \pm 0.48	36.42 \pm 0.46
S	39.32 \pm 0.26	19.63 \pm 0.24	36.51 \pm 0.26
SP	40.34 \pm 0.26	20.38 \pm 0.25	37.52 \pm 0.25
M-SP	40.10 \pm 0.25	20.21 \pm 0.26	37.30 \pm 0.26

Table 6.1: ROUGE F1 scores on the combined AFP/APW/XIN/NYT in-domain test set.

S model, supporting the conclusion that the M-SP model is capable of performing on-the-fly adaptation of output style. This conclusion is further strengthened by the style probability distributions shown in Fig 6.5: they indicate that, for the out-of-domain CNA style, the output mixture is heavily weighted towards the XIN style (0.6 of the probability mass), while for the LTW style, the output mixture weights heavily the NYT style (0.72 of the probability mass). This result is likely to reflect true style characteristics shared by these publishers, since both CNA and XIN are produced by Chinese news agencies (from Taiwan and mainland China, respectively), while both LTW and NYT are U.S. news agencies owned by the same media corporation.

	CNA Test			LTW Test		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
S	40.73 \pm 0.21	17.75 \pm 0.18	37.70 \pm 0.20	27.08 \pm 0.19	8.97 \pm 0.15	25.01 \pm 0.17
M-SP	42.00 \pm 0.20	19.48 \pm 0.21	39.24 \pm 0.22	27.79 \pm 0.19	9.31 \pm 0.18	25.60 \pm 0.17

Table 6.2: ROUGE F1 scores on out-of-domain style test sets CNA and LTW.

6.4.3 Experiment Variants

Model capacity

In order to remove the possibility that the improved performance of the SP model is due simply to an increased model size compared to the S model, we perform an experiment in which we triple the size of the GRU cell dimensions for

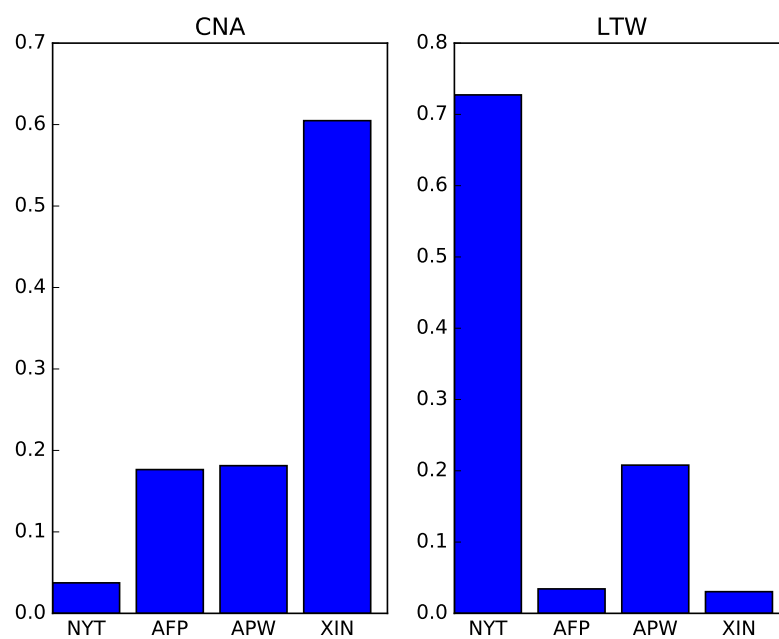


Figure 6.5: Estimated style probabilities over the four in-domain styles, for out-of-domain styles CNA and LTW.

the S model. However, we find no significant performance difference compared to the original dimensions (the ROUGE-L score of the triple-size S model is 36.61, compared to 36.51 obtained of the original S model).

Style embedding

Another approach to separating different styles is to directly encode the style information into the embedding space, in a manner similar to the one proposed in (Johnson et al., 2016). Specifically, at training time, we convert the style label into a one-hot vector and concatenate it with the word embedding at each time step. The ROUGE-L of this model is 36.68, slightly higher than the baseline S model, but significantly lower than the SP model performance (37.52 ROUGE-L). Moreover, we note here that this method is not applicable when the style is either unknown or out-of-domain during testing.

Style vs. Content

Previous experiments indicate that the SP and M-SP models have superior generation accuracy, but it is unclear to what extent the difference comes from improved modeling of style versus modeling of content. To clarify this issue, we performed an experiment in which we replace the named entities appearing in both article and headline with corresponding entity tags, in effect suppressing almost completely any content signal. For instance, given an input such as “China called Thursday on the parties involved in talks on North Korea’s nuclear program to show flexibility as a deadline for implementing the first steps of a breakthrough deal approached.”, paired with goldtruth output “China urges flexibility as NKorea deadline approaches”, we replaced the named entities with their types, and obtained: “LOC_0 called Thursday on the ORG_0 involved in NON_2 on LOC_1 ’s NON_3 to show NON_0 as a NON_1 for implementing the first NON_4 of a NON_5 approached .”, paired with “LOC_0 urges NON_0 as LOC_1 NON_1 approaches.”

Under this experimental conditions, both the SP and M-SP models still achieve significantly better performance compared to the S baseline. For instance,

on the combined AFP/APW/XIN/NYT in-domain test set, the SP model achieves 61.70 ROUGE-L and M-SP achieves 61.52 ROUGE-L, compared to 60.20 ROUGE-L obtained by the S model. On the CNA/LTW out-of-domain test set, M-SP achieves 60.75 ROUGE-L, compared to 59.47 ROUGE-L by the S model.

6.5 Chapter Summary

In this chapter, we describe new style-adaptation model architectures for text sequence generation tasks, SHAPED and Mix-SHAPED. Both versions are shown to significantly outperform models that are either trained in a manner that ignores style characteristics (and hence exhibit a style-averaging effect in their outputs), or models that are trained single-style. The latter is a particularly interesting result, as a model that is trained (with enough data) on a single-style and evaluated on the same style is expected to exhibit the highest performance; our results show that, even for single-style models trained on over 1M examples, their performance is inferior to the performance of SHAPED models on that particular style.

These results support the conclusion that these architectures exhibit both efficient and effective use in their modeling of both generic language phenomena as well as particular style characteristics.

Chapter 7

Active Discriminative Text Representation Learning

7.1 Chapter Overview

The most straightforward way of overcoming low-supervision issue is to collect more useful training labels. Assuming that annotating data is expensive, there is often a budget on how many labels that we could gather. Under this budget, how to gather more valuable data points to maximize the model performance is worth studying. The process of collecting more helpful data points in an intelligent way is called *active learning* (AL) (Settles, 2012).

In this chapter, we propose an AL method specifically for CNN introduced in Chapter 2 (Zhang et al., 2017a)¹. While CNNs (and neural networks more generally) have demonstrated excellent performance when one has access to large amounts of training data, how can we make the best use of CNNs when annotation resources are scarce? Because word embedding estimation and tuning (for a specific text classification task) may be viewed as *representation learning*, it is reasonable to optimize feature vectors before expending effort to tune the parameters of a model that accepts these as input. Indeed, adjusting the former will render updates to the latter potentially useless. Thus, we argue that the objective in AL (at least at the outset) should primarily be to select instances that result in better representations.

More specifically, we propose a novel AL approach for sentence classification in which we select instances that contain words likely to most affect the embeddings. We achieve this by calculating the expected gradient length (EGL) *with respect to the embeddings* for each word comprising the remaining unlabeled sentences. We show that this approach allows us to rapidly learn discriminative, task-specific embeddings. For example, when classifying the sentiment of sen-

¹Ye Zhang, Matthew Lease and Byron Wallace. 2017. Active Discriminative Text Representation Learning. In *Proceedings of Association for the Advancement of Artificial Intelligence*. I proposed the idea, implemented the idea and wrote the paper.

tences, we find that selecting examples in this way quickly pushes the embeddings of ‘bad’ and ‘good’ apart (Figure 7.2, bottom row). Ultimately, results show our AL method improves accuracy over several baseline AL approaches, across sentence and document classification tasks considered.

This method selects instances based on a max operator over the gradients expected for the individual words in a text, and thus is less appropriate for longer texts such as documents. Therefore, we extend our approach for document classification by linearly combining two scores: one corresponding to individual word embeddings and one measuring the overall uncertainty regarding instances.

In summary, key contributions of this chapter include:

- As far as we are aware, this is the first work to consider AL strategies explicitly for neural architectures in the context of text classification.
- We demonstrate that variants of our model outperform baseline AL approaches that do not consider embedding-level parameters: on both sentence and document classification tasks our method realizes better performance with fewer labels, compared to baseline sampling approaches.
- We also note that our approach substantially reduces the computational cost of AL, compared to previously proposed EGL approaches to AL.

7.2 Prior Work

We consider a *pool-based* AL scenario (Zhu et al., 2008; Tong and Koller, 2001), in which there exists a small set of labeled data \mathcal{L} and a large pool of available unlabeled data \mathcal{U} . The task for the learner is to draw examples to be labeled from \mathcal{U} cleverly, so as to maximize classifier performance. These selections, or *queries*, are typically made in a greedy fashion; an informativeness measure is used to score all candidate instances in the pool, and the instance maximizing this measure is selected.

The key to developing AL strategies is designing a good informativeness measure. Let \mathbf{x}^* be the most informative instance according to a *query strategy* $\phi(\mathbf{x}_i; \boldsymbol{\theta})$, or function used to evaluate each instance \mathbf{x}_i in the unlabeled pool \mathcal{U}

conditioned on the current set of parameter estimates θ . We can define the following instance selection protocol:

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}_i \in \mathcal{U}} \phi(\mathbf{x}_i; \theta) \quad (7.1)$$

For CNNs, θ includes word embedding parameters \mathbf{E} , convolution layer parameters \mathbf{C} , and softmax layer parameters \mathbf{W} .

Many querying strategies have been proposed in the literature (Settles, 2010). Our aim here is to ascertain whether AL works better in the case of neural models when one explicitly considers representation learning (i.e., focussing on \mathbf{E}); we selected the following three general baseline approaches because they enable us to explore this question directly.

Random sampling. This strategy is equivalent to standard (or ‘passive’) learning; here the training data is simply an i.i.d. sample from \mathcal{U} .

Uncertainty sampling. Perhaps the most commonly used query strategy is *uncertainty sampling* (Lewis and Gale, 1994; Tong and Koller, 2002; Zhu et al., 2008; Ramirez-Loaiza et al., 2016), in which the learner requests labels for instances about which it is least certain *wrt.* categorization.

Uncertainty sampling can be instantiated in many ways, depending on the underlying classification model. A general uncertainty sampling variant uses entropy (Shannon, 1948) as an uncertainty measure, defining $\phi(\mathbf{x}_i; \theta)$ as:

$$-\sum_k P(y_i = k | \mathbf{x}_i; \theta) \log P(y_i = k | \mathbf{x}_i; \theta) \quad (7.2)$$

where k indexes all possible labels. Entropy-based uncertainty sampling often performs well (Settles, 2010).

Expected Gradient Length (EGL). This AL strategy aims to select instances expected to result in the greatest change to the current model parameter estimates when their labels are revealed (or provided) (Settles and Craven, 2008). The intuition is that one can view the magnitude of the resultant gradient as the value of purchasing a label; if this cost is small, then the label did not provide much new

information. If the true class for a given instance were known, the gradient could be directly calculated under this assignment. But in practice this is unknown, and so the expectation is taken by marginalizing over the gradients calculated conditioned on possible class assignments, scaled by current model estimates of the posterior probabilities of said assignments.

7.3 Method

We now introduce our proposed AL strategy for text classification with embeddings. This is based on the EGL method described above. In gradient-based optimization for neural models, the training gradient back-propagated to a set of model parameters given label y_i for instance \mathbf{x}_i may be viewed as a measure of change imparted by example i for those parameters. Thus the learner should request the label for an instance expected to produce a large magnitude training gradient. If this gradient is taken with respect to all model parameters (distributed over all layers), then this is a straight-forward instantiation of EGL. Past work on EGL (involving linear models) adopted exactly this approach: the expected change to model parameters was evaluated over the entire set of parameters in θ . By contrast, we propose *explicitly selecting examples that are likely to affect the representation-level parameters (i.e., the embeddings)*.

Formally, let $\nabla J(\mathcal{L}; \theta)$ be the gradient of the objective function J with respect to the model parameters θ , where J is the cost function. Further, let $\nabla J(\mathcal{L} \cup \langle \mathbf{x}_i, y_i \rangle; \theta)$ be the new gradient that would be obtained by adding the training tuple $\langle \mathbf{x}_i, y_i \rangle$ to \mathcal{L} . Because the true label y_i will be unknown, we take an expectation over possible class assignments k . More precisely, we can calculate $\phi(\mathbf{x}_i; \theta)$ as:

$$\sum_k P(y_i = k | \mathbf{x}_i; \theta) \|\nabla J(\mathcal{L} \cup \langle \mathbf{x}_i, y_i = k \rangle; \theta)\| \quad (7.3)$$

where $\|\mathbf{r}\|$ denotes the Euclidean norm of \mathbf{r} . Note that at query time $\|\nabla J(\mathcal{L}; \theta)\|$ should be near zero, assuming J converged during the previous iteration. Thus, we can approximate $\nabla J(\mathcal{L} \cup \langle \mathbf{x}_i, y_i = k \rangle; \theta) \approx \nabla J(\langle \mathbf{x}_i, y_i = k \rangle; \theta)$ for efficiency.

This approach selects instances that are likely to most perturb all model parameters θ . However, ‘deep’ neural architectures are distinguished by their multi-layered structure, which corresponds to a large set of features distributed across different layers in the architecture. This makes calculating the EGL computationally expensive. More importantly, it is arguably incoherent to jointly consider the expected change *at different layers* in the model. If we view lower levels in the model as learning to extract features, it makes little sense to jointly maximize expected change in these features *and* to the parameters of the final softmax layer that accepts these as input. Changes to the former will immediately change the implications of perturbing the latter.

Instead, we want to select unlabeled instances that can most improve the features learned by the model. Intuitively, it is paramount that the model learn good (discriminative) *representations*; these will feed forward through the network, in turn improving classification. In the context of sentence classification — in which instances comprise relatively few words — we propose a querying strategy that scores sentences using the maximum expected gradient over the words they contain. In the case of longer texts or documents (which contain many words), it is intuitive to strike a balance between myopically selecting instances to maximize individual word gradients on the one hand, and considering the model’s overall uncertainty regarding the instance on the other. We next elaborate on the methods we propose for these two scenarios.

7.3.1 Active Sentence Classification with CNNs

EGL-word model. For sentence classification, we adopt the following as our scoring function for sentence classification. For each instance (sentence) in \mathcal{U} , we take the expected gradient with respect to *only* the embeddings of its constituent words, selecting the example that maximizes this expected embedding gradient as our measure of informativeness. Intuitively, we use a max-over-words approach to adjust particular word embeddings that are discriminative for the task at hand. Formally,

we define our $\phi(\mathbf{x}_i; \boldsymbol{\theta})$ as:

$$\max_{j \in \mathbf{x}_i} \sum_k P(y_i = k | \mathbf{x}_i; \boldsymbol{\theta}) \|\nabla J_{\mathbf{E}(j)}(\langle \mathbf{x}_i, y_i = k \rangle; \boldsymbol{\theta})\| \quad (7.4)$$

Where we denote by $\nabla J_{\mathbf{E}(j)}$ the gradient of J with respect to the embedding of word j (j ranges over the words in \mathbf{x}_i). Note that the gradient is only taken for each word in the instance \mathbf{x}_i . The gradients for embeddings corresponding to words not in \mathbf{x}_i are 0 and can thus be ignored; this is a computational boon because instances tend to be sparse. Another straightforward strategy to measure the informative of a sentence is to replace the ‘max’ operator in equation 7.4 with the average operation. That is, instead of choosing the word with the maximum expected gradient, we can average on the expected gradients of all the words in the sentence. But this method does not work as well as *EGL-word*. We attribute this to the fact that in a short sentence, most words are not relevant to the label of the sentence.

***EGL-sm* model.** Whereas *EGL-word* focuses on parameters associated with the lowest level in the model, we also consider the other extreme in sentence classification tasks: taking the gradient with respect to only the final softmax layer parameters \mathbf{W} . In this case $\phi(\mathbf{x}_i; \boldsymbol{\theta})$ becomes:

$$\sum_k P(y_i = k | \mathbf{x}_i; \boldsymbol{\theta}) \|\nabla J_{\mathbf{W}}(\langle \mathbf{x}_i, y_i = k \rangle; \boldsymbol{\theta})\| \quad (7.5)$$

where $J_{\mathbf{W}}$ denotes the gradient *wrt.* the softmax layer.

7.3.2 Active Document Classification with CNNs

***EGL-word-doc* model.** For longer text classification tasks, we modify the above *EGL-word* variant in a few key ways. First, we normalize the gradient of each word by dividing it by its frequency in the document. This is because in longer texts there exist many ‘stop words’ such as ‘the’, and their gradients dominate if occurrence counts are ignored, since there are more branches flowing back to these words during back-propagation. Accounting for term frequencies in the gradient calculation mitigates this issue. Second, rather than exclusively relying on the single word with

the largest gradient to score documents, we sum over the (frequency-normalized) gradients corresponding to the top k words. The number of top words (k) is a hyper-parameter and will depend on the average document length in a given corpus. We refer to this method as *EGL-word-doc* for document classification.²

***EGL-Entropy-Beta* model.** In addition to the above modifications, we extend our approach for longer text classification to jointly consider: (1) the expected updates to word gradients (for words in the instance); and (2) the current uncertainty regarding the instance. For the former, we use *EGL-word-doc* (modified as described above), and for the latter we use entropy (Equation 7.2). We denote the entropy score by ϕ_{Entropy} and the *EGL-word-doc* score by $\phi_{\text{EGL-word-doc}}$. We interpolate these to form a composite document score.

These scores are on incomparable scales, so we normalize them by transforming them into percentiles. $\mathcal{P}(i, \mathcal{U})$ is used to denote the percentile of the score of a given instance among a pool of instances \mathcal{U} . For example, $\mathcal{P}(i, \mathcal{U})=87\%$ indicates that 87% of the instances in \mathcal{U} are smaller than i . To encode the relative entropy score of a given instance in \mathcal{U} , we use $\mathcal{P}(\phi_{\text{Entropy}}(i), \{\phi_{\text{Entropy}}(j) : j \in \mathcal{U}\})$. We can now define our composite, interpolated scoring function which considers feature learning and output certainty jointly:

$$\begin{aligned} \phi_t(i) = & \gamma_t \cdot \mathcal{P}(\phi_{\text{Entropy}}(i), \{\phi_{\text{Entropy}}(j) : j \in \mathcal{U}\}) + \\ & (1 - \gamma_t) \cdot \mathcal{P}(\phi_{\text{EGL-word-doc}}(i), \{\phi_{\text{EGL-word-doc}}(j) : j \in \mathcal{U}\}) \end{aligned} \quad (7.6)$$

We treat the interpolation parameter γ_t — constrained to be between 0 and 1 — as a random variable with a temporal dependence (t indexes time, or AL iteration). Intuitively, we assume that at the outset of AL, the model should pay relatively more attention to learning discriminative representations of words. As learning progresses, focus should shift toward the higher-level uncertainty-based

²Experiments applying the same variant of *EGL-word* used for sentence classification does not perform as well for longer texts. *EGL-sm* model also performs much worse than the other methods in the document classification tasks, so we do not report their results.

score. To realize this intuition, we assume $\gamma_t \sim \text{Beta}(\alpha, \beta_t)$. We decrease β_t linearly over time (AL iterations), which has the desired effect of increasing the expectation of γ_t , in turn increasing the attention paid to the document level entropy score. We found that drawing γ_t from a distribution yields smoother performance compared to setting it deterministically.

7.4 Experiment

We report results on three sentence datasets and three document datasets. Tables 7.1 and 7.2 provide key statistics for each dataset. We briefly describe each dataset below and refer the reader to the source citations for additional details.

	CR	MR	Subj
Pos / Neg	2406 / 1367	5331 / 5331	5000 / 5000
Avg. word count	19	20	23

Table 7.1: Statistics of sentence datasets.

	MR	MuR	DR
Pos / Neg	1000 / 1000	1000 / 1000	23649 / 30254
l_{sen}	21.2	16.8	15.2
l_{doc}	32.6	7.5	4.1

Table 7.2: Statistics of document datasets. l_{doc} denotes the average sentence length in words, and l_{doc} denotes the average number of sentences per document.

Sentence Datasets

CR: *positive / negative* product reviews (Hu and Liu, 2004).³

MR: *positive / negative* movie reviews (Pang and Lee, 2005b).

Subj: *subjective / objective* sentences (Pang and Lee, 2004).⁴

³www.cs.uic.edu/liub/FBS/sentiment-analysis.html

⁴MR and Subj datasets are available at: <http://www.cs.cornell.edu/people/pabo/movie-review-data/>.

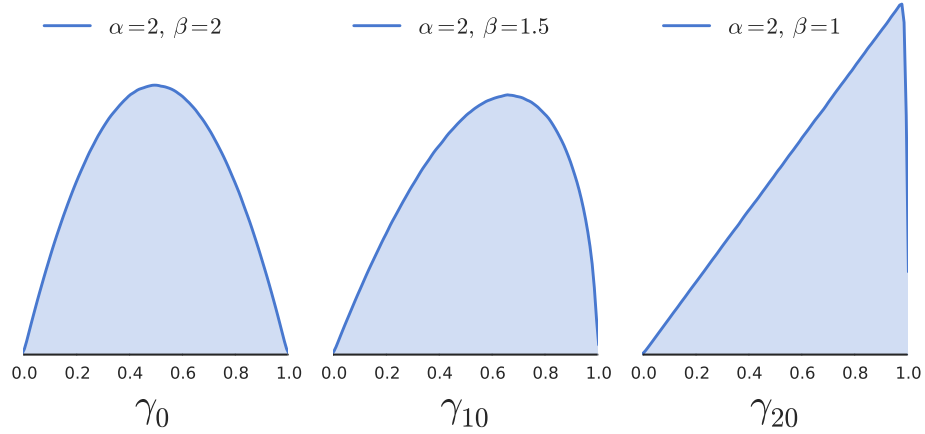


Figure 7.1: Beta distributions over γ_t at $t=0, t=10, t=20$.

Document Datasets *positive / negative* classification tasks

MR: (Longer) movie reviews (Pang and Lee, 2004)⁵.

MuR: Music reviews (Blitzer et al., 2007).⁶

DR: Doctor reviews (Wallace et al., 2014b).

7.4.1 Model Configuration

We used standard pre-trained `word2vec`-induced vectors⁷ to initialize **E**. As per Zhang and Wallace (Zhang and Wallace, 2015a), we used three filter heights (3, 4, 5). For sentence and document classification tasks, we used 50 and 100 filters of each size, respectively.⁸ Given our goal to explore AL strategies appropriate for neural architectures (particularly CNNs), rather than to maximize absolute CNN performance for new state-of-art results, we did not tune these hyperparameters.

We performed 20 rounds of batch active learning. At the outset, we provided all learners with the same 25 instances (sampled i.i.d. at random). In subsequent

⁵Both MR datasets can be found online at the same URL.

⁶<http://www.cs.jhu.edu/~mdredze/datasets/sentiment/>

⁷<https://code.google.com/archive/p/word2vec/>

⁸We used more filters for document classification tasks because we expect more diversity in longer pieces of text, but we found that the performance was not sensitive to this choice in any case.

rounds, each learner was allowed to select 25 instances from \mathcal{U} according to their respective querying strategies. These examples were added to \mathcal{L} , and the models were retrained.

For *EGL-word-doc* and *EGL-Entropy-Beta* in document classification, the number of top words k used to calculate the score for each document was set to 3, 2 and 30 respectively for MuR, DR, and MR datasets. For *EGL-Entropy-Beta*, we fixed $\alpha = 2$ and initialized $\beta = 2$ as well, which implies a roughly equal weight on embedding and uncertainty scores. We then decreased β_t linearly with iterations t . Thus γ_t is expected to increase over time, ascribing more weight to the entropy score. For reference, Figure 7.1 provides illustrative empirical distributions used for γ at three time points during AL. To reiterate, our goal was to shift from initially paying equal attention to the representation learning and instance uncertainty criteria, to increasingly focusing on the latter (document-level uncertainty) as time progresses.

We evaluated performance by calculating accuracy (classes are fairly balanced) on a held-out test set after each round. For all but one dataset we repeated this entire AL process 10 times, using test sets generated via 10-fold CV. The exception was the doctor reviews (‘DR’) dataset, which is comparatively large; we therefore used a single big test set in this case. We replicated all experiments 5 times for all train/test splits, for all datasets, to account for variance. We estimated parameters by Adadelta (Zeiler, 2012a), tuning \mathbf{E} in back-propagation to induce discriminative embeddings.

7.5 Results and Discussion

We now report results. For sentence classification, we use the simple variant of our method (*EGL-word*) which is more appropriate for short texts (since it is ultimately a max-operator over expected gradients for individual words). For document classification, we also use the interpolated method, which considers expected gains both with respect to feature learning and in terms of instance-level uncertainty reduction. This method is more appropriate for longer texts.

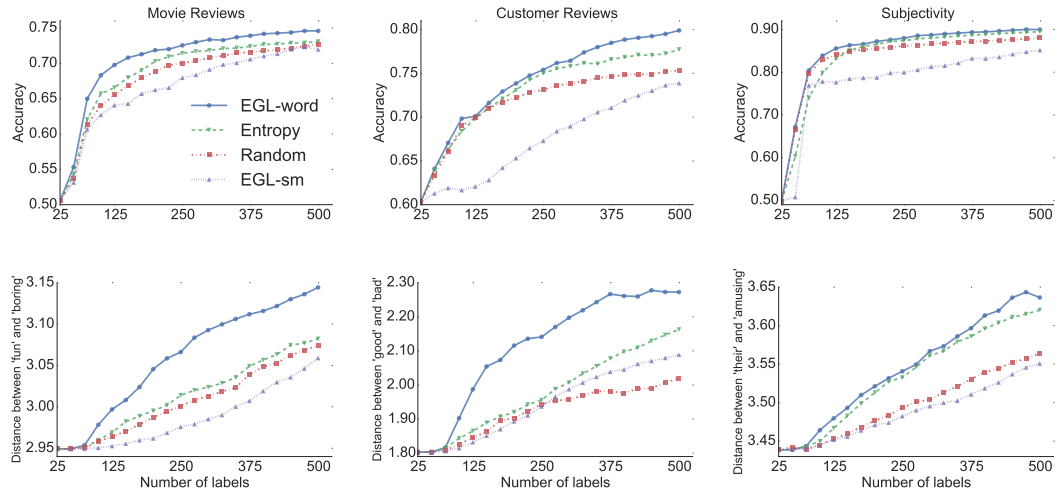


Figure 7.2: Results on the three sentence classification datasets. **Top row:** number of labels versus accuracy. **Bottom row:** number of labels versus the distance between tuned embeddings for selected pairs of informative words (with opposite polarity) for each dataset. The scale in this case, which captures the Euclidean distance in the embedding space, has only relative meaning.

7.5.1 Sentence Classification Results

Figure 7.2 reports learning curves on the three sentence datasets. The proposed *EGL-word* active learning method outperforms baseline approaches, performing especially well on sentiment analysis tasks (MR and CR). We believe this is due to our model rapidly learning more discriminative representations of words with opposing polarities.

To further illustrate this point, Figure 7.2’s bottom row provides plots displaying the Euclidean distances between selected pairs of word embeddings induced using different AL strategies. In the customer review (CR) dataset, for example, we consider the embeddings of words ‘good’ vs. ‘bad’ and see that *EGL-word* quickly pushes these embeddings apart. Similarly, on the movie review (MR) dataset, ‘fun’ and ‘boring’ are rapidly separated in embedding space. The subjectivity (Subj) detection task is less clear-cut. Here we picked words ‘amusing’ and ‘their’, because ‘amusing’ strongly indicates subjectivity, while ‘their’ is plainly

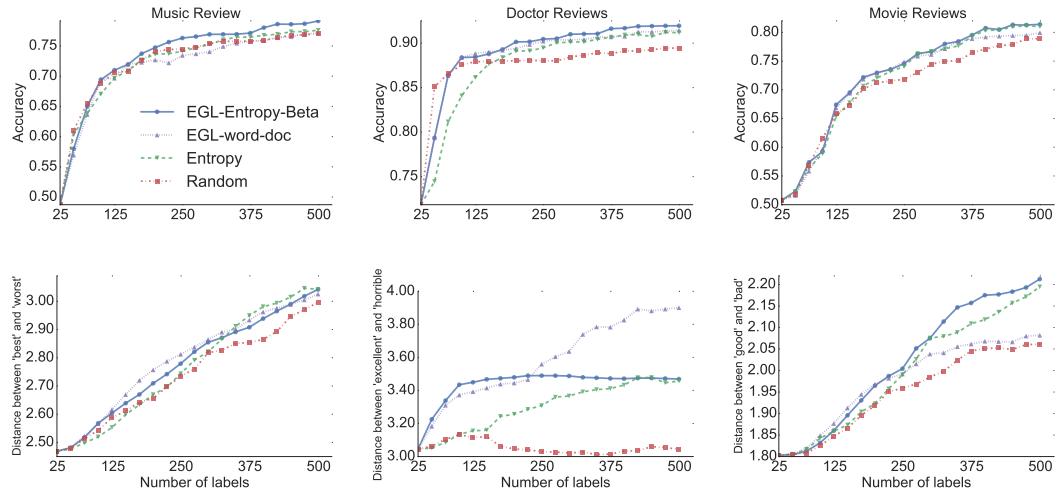


Figure 7.3: Results on the three document datasets. **Top row:** number of labels versus accuracy. **Bottom row:** number of labels versus the distance between tuned embeddings for selected pairs of informative words (with opposite polarity) in each task.

Methods that explicitly consider representation/embedding parameters more quickly push discriminative word vectors apart. Intuitively, the distances between the contrasting word-pairs increases quickly with both of the proposed EGL methods. However, recall that the *EGL-Entropy-Beta* method differs from *EGL-word-doc* in interpolating entropy along with expected updates to word gradients. As a result, we observe that *EGL-Entropy-Beta* method tends to shift from rising with *EGL-word-doc* at the start of learning, while later merging with the distances achieved by the *Entropy* method as learning progresses. This transition corresponds to first focusing on embeddings, and then later shifting emphasis to the entropy criterion.

neutral. As expected, *EGL-word* quickly pushes these apart, though less rapidly than with the sentiment tasks.

Table 7.3 reports Area Under Curve (AUC) scores for each learning curve from 25-500 labeled instances using trapezoidal rule (Süli and Mayers, 2003). We normalize AUC by the maximum possible for the range: $(500 - 25) * 1 = 475$.

	EGL-word	Entropy	Random	EGL-sm
MR	0.707	0.690	0.681	0.667
CR	0.743	0.732	0.720	0.674
Subj	0.856	0.840	0.839	0.785

Table 7.3: Area Under (learning) Curve (AUC) scores on sentence classification datasets; bold indicates best results.

	E-E-B	EGL-word-doc	Entropy	Random
MR	0.725	0.719	0.719	0.704
DR	0.893	0.889	0.877	0.878
MuR	0.736	0.718	0.725	0.726

Table 7.4: Area Under (learning) Curves (AUC) scores on the three document datasets. E-E-B refers to *EGL-Entropy-Beta*.

7.5.2 Document classification results

Figure 7.3 displays learning curves achieved on the document classification datasets, and Table 7.4 reports the corresponding AUC scores achieved by each method on each dataset. Overall, the *EGL-Entropy-Beta* outperforms other methods, demonstrating the value of explicitly selecting examples likely to improve representation level parameters.

Results using the simple variant of *EGL-word-doc* are mixed. In general it outperforms baselines only during the first several iterations of AL, but is later outperformed by entropy-based sampling. Our intuition here is that narrowly focusing on improving feature representations provides early gains, but longer texts require attention to be shifted to instance-level uncertainty. And indeed, the proposed *EGL-Entropy-Beta* method consistently performs more robustly, and tends to realize the best of both worlds, achieving rapid gains but also generally maintaining dominance over all AL iterations.

Similar to Figure 7.2’s bottom row for sentence tasks, Figure 7.3’s bottom row shows for document tasks how distances between selected word embeddings grow as more examples are collected. *EGL-word-doc* and *EGL-Entropy-Beta* consistently push the representations for the selected polar word-pairs apart more

rapidly than other methods. However, recall that the *EGL-Entropy-Beta* method differs from *EGL-word-doc* in interpolating entropy along with expected updates to word gradients. As a result, we observe that *EGL-Entropy-Beta* method tends to shift from rising with *EGL-word-doc* at the start of learning, while later merging with the distances achieved by the *Entropy* method as learning progresses. *EGL-Entropy-Beta* thus strikes a balance between this and refining the parameters at higher levels in the model, as evidenced by the superior classification performance seen in the top row of Figure 7.3. Maintaining a narrow focus on embeddings only ultimately results in comparatively poor performance in the case of document classification.

7.6 Chapter Summary

The importance of *representation learning* (Bengio, 2009) with neural models motivates exploring new, representation-based active learning (AL) approaches with neural models. In this chapter, we proposed a new AL strategy for CNNs that is specifically designed to quickly induce discriminative, task-specific representations (word embeddings), thus improving classification. We showed that this approach outperforms baseline AL strategies across sentence and document classification datasets considered, and that such discriminative word embeddings can be rapidly induced.

We believe that these encouraging results will help to stimulate further research on active learning tailored to deep/hierarchical architectures. Our own future work will include generalize the similar AL strategies to other neural models such as recurrent neural network and improving the modeling strategy for γ_t (the parameter governing relative emphasis on representation vs. instance-level uncertainty), perhaps based on reinforcement learning. We also envision augmenting the model to optimize instance selection in terms of refining additional intermediate layer representations in deeper networks.

Chapter 8

Active Transfer Learning for Neural Models

8.1 Chapter Overview

We investigate transfer learning and domain adaptation in Chapter 5 and Chapter 6 respectively, and then active learning in Chapter 7. It is then very natural to consider combining transfer learning and active learning, that is, we are allowed to gather some amount of labels from the target domain but at the same time, we can train a model on a source domain which has large amount of labels. Suppose that we could transfer the source domain model to the target domain, how to do active learning in this situation?

In this chapter, we consider active transfer learning that does active learning in the target domain while at the same time can adapt the model trained on the source domain¹. Intuitively, this may be accomplished by selecting for annotation points in the target domain that are maximally misaligned with the supervision in the source domain. We utilize the recently proposed influence function (Koh and Liang, 2017) to achieve this. The influence function is originally used for explaining model prediction by discovering the most influential training point. We take a closer examination of its use for explaining neural NLP models.

Contributions. 1) To the best of our knowledge, this is the first work to investigate the use of influence functions for black-box (neural) models in NLP. As many NLP tasks are structured, we extend the approach to accommodate this. 2) We propose a novel active transfer learning strategy that uses the influence function to identify candidate instances in the target domain for labeling.

Though our method is novel and interesting, however, empirically, we find that it achieves very mixed results. So we do not think this method is applicable in practice. Further thoughts are needed to make this method work.

¹This chapter's work is unpublished.

8.2 Prior Work

There is a body of emerging work on interpreting the inner workings of neural NLP models. Karpathy et al. (2015) performed an in-depth empirical investigation of character-based LSTM language models and performed error analysis to shed insight into the workings and limitations of RNNs. Elsewhere, Li et al. (2015) use ‘saliency heatmaps’ to interpret the contribution to the final disposition attributable to the respective ‘compositional units’ (e.g., words). Li et al. (2016a) subsequently used ‘feature erasion’ to highlight the contribution of individual model components to the overall score. These efforts share the high-level aim of attempting to explain why the model has come to a particular prediction for a given input. A slightly different tact was proposed by Alvarez-Melis and Jaakkola (2017): they aim to discover the dependencies between tokens in structured input-output pair of neural models by modeling this as partitioning problem.

In terms of active transfer learning, there is some work on active transfer (e.g., (Rai et al., 2010)). Wu et al. (2017) and Li et al. (2012) propose active transfer learning methods specifically for linear models. There is work on transferring neural NLP models (Yang et al., 2017; Mou et al., 2016). Recent work has also investigated which source domain examples are transferable to a target domain (Ruder and Plank, 2017). This differs from our work in the objective; they attempted to select source points to transfer, rather than actively select points in the target domain. However, in this work we have proposed an approach for active transfer of neural NLP models.

8.3 Influence Function and Explaining Predictions in NLP

Consider a standard supervised machine learning setting in which the goal is to map from a given input space \mathcal{X} to a target output space \mathcal{Y} . We assume access to training points z_1, z_2, \dots, z_N , where $z_i = (x_i, y_i)$, $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. During model training, the objective is to minimize the empirical loss $\frac{1}{N} \sum_{i=1}^N \mathcal{L}(z_i, \theta)$, where $\mathcal{L}(z, \theta)$ is the loss for point z , and θ are model parameters. The empirical

loss minimizer is defined as $\hat{\theta} = \operatorname{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(z_i, \theta)$.

Given fitted model parameters $\hat{\theta}$, we can make a prediction for a test point z_{test} and calculate a corresponding loss $\mathcal{L}(z_{\text{test}}, \hat{\theta})$. One means of realizing post-hoc model interpretability of the *explanation by example* variety (Lipton, 2016) is to identify the training point that most influenced the prediction for a particular test point z_{test} . This training example may be characterized as either helpful (when it nudges the model toward the correct prediction for z_{test}) or harmful (when it does the opposite). A naive approach to identifying such points would be to remove each training point in turn and retrain the model, observing any induced changes in loss on z_{test} . But this method is hopelessly inefficient.

Koh and Liang (2017) recently proposed using the *influence function* to provide a closed-form derivation of the change in loss w.r.t. a test point z_{test} that would be induced, were one to increase z_i 's training instance weight prior to retraining²:

$$I(z_i, z_{\text{test}}) = \nabla_{\theta} \mathcal{L}(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_{\theta} \mathcal{L}(z_i, \hat{\theta}) \quad (8.1)$$

Where $\mathcal{L}(z_i, \hat{\theta})$ is the loss function of training point z_i under the empirical loss minimizer $\hat{\theta}$, ∇ denotes the gradient, and $H_{\hat{\theta}}$ is the Hessian matrix: $\frac{1}{N} \sum_{i=1}^N \nabla_{\theta}^2 \mathcal{L}(z_i, \hat{\theta})$. A large influence function value implies that upweighting z_i would decrease $\mathcal{L}(z_{\text{test}}, \hat{\theta})$. Thus the larger $I(z_i, z_{\text{test}})$, the more helpful z_i w.r.t. z_{test} ; a large magnitude negative value implies a harmful training instance.

To efficiently calculate the inverse Hessian vector product in Eq. 8.1, the conjugate gradient method (Martens, 2010) can be used.

Influence functions provide machinery for interpreting black-box models (including deep neural networks), but they were originally proposed for application to image data, and only tangentially applied to text (Koh and Liang, 2017). It is not immediately clear how well the approach might generalize to problems in language. Will ‘influential’ training texts be meaningful (or useful)?

In this section, we report experiments investigating the applicability and util-

²To ease interpretation we have defined the influence function here as the *negative* of the derivative of the test loss w.r.t. changes in the weighting of z_i ; thus a large influence function value implies a helpful training example. In the original derivation (Koh and Liang, 2017), this was reversed.

	Train	Test	AVG	LSTM
MR	9555	1107	75.88	78.59

Table 8.1: Sentence count in the movie review (MR) sentiment dataset, and test accuracy (%) achieved by AVG and LSTM models on it.

ity of influence functions for text classification and sequence tagging . Rather than improving state-of-the-art predictive performance on these tasks, here *our goal is to illustrate the influence function as a means of explaining the predictions of arbitrary neural models*. Our results suggest that influence functions are indeed useful for language tasks and may facilitate inspection and debugging of neural NLP models. In Section 8.4 we introduce a novel active transfer learning strategy that uses the influence function to identify domain mismatch between source and target instances.

8.3.1 Text Classification

We consider binary (sentiment) classification, selecting two models that are representative of modern neural architectures. The first is a simple continuous bag-of-words (CBOW) approach, in which we average the embeddings of the words comprising a text. This aggregate vector is then fed to a softmax layer. We will refer to this model as ‘AVG’. For a second model, we pass a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) (using a hidden state vector dimensionality of 64) over texts, feeding the hidden state output from the last time step to a softmax layer.

Next, we experiment with the use of influence functions on a simple sentiment classification task, using a corpus of movie reviews (MR) (Pang et al., 2002). For this we manually set aside 1/10th of the data for testing.³ Table 8.1 reports dataset statistics and the accuracy achieved by AVG and LSTM models. Note that the class distribution here is balanced. For each test instance, we can use Eq. 8.1 to calculate the influence of each training point w.r.t. test predictions (e.g., to identify

³We randomly select $\sim 20\%$ of the training data to be used as the development set; this is used to inform early stopping.

Test sentence (classified as *negative*; true label *positive*): Narc may not get an ‘A’ for originality, but it wears its b-movie heritage like a badge of honor.

Most harmful training sentence (LSTM) (label *negative*): The original wasn’t a good movie but this remake makes it look like a masterpiece!

Nearest neighbor (LSTM) (label *negative*): The locale ... remains far more interesting than the story at hand.

Table 8.2: A misclassified movie review (MR) sentence and corresponding ‘most harmful’ training point, as identified via the influence function.

helpful and harmful training points).

Another straightforward way of finding related and potentially ‘responsible’ training points would be to retrieve the nearest neighbors of the test point in the learned embedding space that precedes the output layer (e.g., using the Euclidean distance). In AVG, this is the CBOW embedding of an instance. In the LSTM model, this is the output at the last time step (the penultimate layer). Koh and Liang (2017) demonstrated that for images, points identified as responsible for the predictions made for test instances via the influence function differ from those that are simply nearby in the embedding space, i.e., *influence* differs from mere *similarity*. Does this hold for language data?

To investigate, we plot influence function values against Euclidean distances in Figure 8.1. For this analysis, we randomly select one correctly and one incorrectly classified test point (movie review). Figure 8.1a shows the former, and Figure 8.1b shows the latter. The true label of both reviews was positive. For the LSTM, we plot influence function values against Euclidean distances between the test and the training points, using the final hidden state representations induced at the penultimate layer of the LSTM (left column of Figure). For the AVG model, we plot the influence function vs. the distance to each training point under CBOW representations (right column of Figure). In all subplots, blue x’s are positive and yellow triangles are negative.

We observe in Figure 8.1 that in the simple AVG model, training points far away from the test points in CBOW space exert almost zero influence (right

column). This implies that AVG behaves much like a nearest neighbor method. By contrast, in the LSTM there is less correlation between influence and distance: training points far away (in the induced representation extracted from the penultimate layer) can still be influential, and many nearby points impart little to no influence (left column). A second observation is that in AVG, positive training points close to the test point are uniformly helpful, and negative training points close to the test point are all harmful, regardless of whether the prediction is correct or not (right column). In contrast, when the LSTM makes a correct prediction, some negative training points are helpful, and when LSTM makes a misclassification, some positive training points are harmful.

To illustrate use of the influence function for debugging neural text classification models, Table 8.2 shows an incorrectly classified positive movie review alongside its (1) most harmful training point, identified via the influence function, and (2) nearest neighbor in the training set (both w.r.t. the LSTM model). The harmful sentence found via the influence function suggests that the model may have erroneously learned to categorize instances of the sentence pattern *negative + but + positive* as negative, suppressing the positive sentiment after *but*. The harmful sentence differs substantially from simply taking the nearest neighbor. The influence function may allow us to pick up on subtle causes of misclassifications.

	Train	Dev	Test
CoNLL	14987	3466	3684
Twitter	1900	240	254

Table 8.3: Number of sentences in the train, dev and test splits of the CoNLL 2003 and Twitter NER corpora used in sequence tagging experiments.

8.3.2 Sequence Tagging

We now turn to debugging structured (sequence tagging) models. For this we use character-augmented bi-directional LSTM (Lample et al., 2016) on two datasets: (1) the CoNLL 2003 named entity recognition (NER) corpus (Tjong Kim Sang and De Meulder, 2003); and (2) a Twitter NER dataset (Ritter et al.,

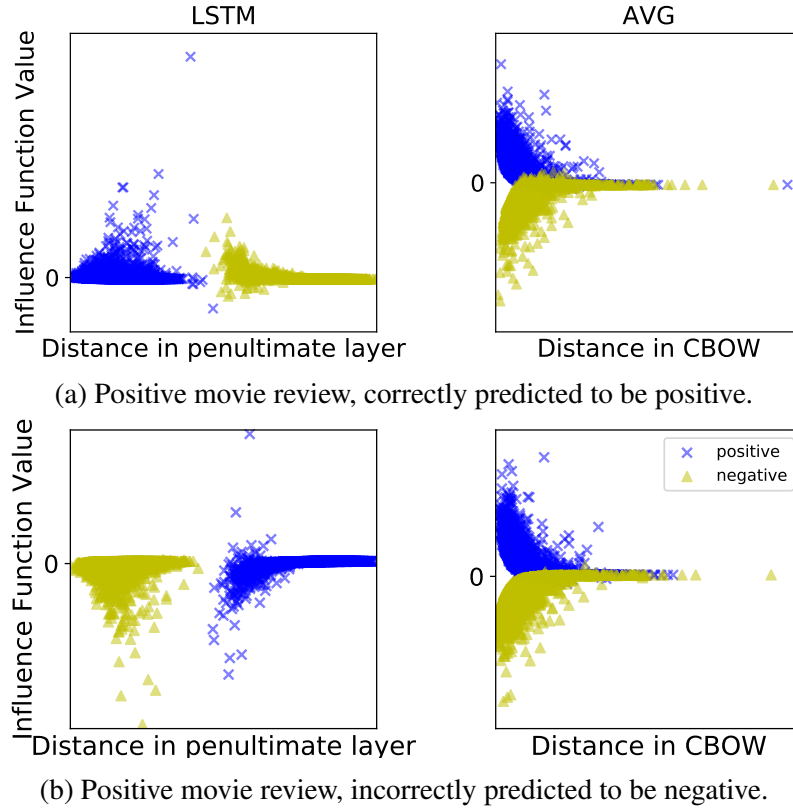


Figure 8.1: The relationship between influence function values and distances between embedded instances. For test point z_{test} , we show a scatter plot of the influence function of each training point w.r.t. z_{test} against the Euclidean distances between these. Blue \times 's and yellow \triangle 's denote positive and negative training points, respectively.

2011). The former has a standard train/dev/test split, and we created analogous splits for the latter. We report corpus statistics in Table 8.3. CoNLL includes four entity types: *location* (LOC), *person* (PER), *organization* (ORG), and *miscellaneous* (MISC). The Twitter corpus defines ten types: *person*, *geo-location*, *facility*, *company*, *product*, *tv show*, *film*, *music artist*, *sports team*, and *other*. For both, we follow the beginning (B), inside (I) and outside (O) tagging scheme, in which ‘B’, and ‘I’ denote tokens that begin and continue entity spans, and ‘O’ indicates that a token is outside of an entity.

Test instance: Jaegal (<i>B-PER, B-PER</i>) Sung-Yeol (<i>I-MISC, I-PER</i>) (South Korea) 37.46; 3.
Harmful training instance: Millns (<i>B-MISC</i>), who toured Australia with England A in 1992/93 , replaces former England all-rounder Phillip DeFreitas as Boland ’s overseas professional.
Test instance: So far this year Zywiec (<i>B-LOC, B-ORG</i>), whose full name is Zakłady Piwowarskie w Zywcu SA, has netted six million zlotys on sales of 224 million zlotys .
Harmful training instance: Index heavyweights Elf and Rhone Poulenc both ended slightly weaker while active Eurotunnel (<i>B-LOC</i>) was unchanged on nearly a million shares traded .
Test instance: Trade and Industry Secretary Ian Lang added that even if the conditions were met by both airlines, final clearance would hinge on an open skies deal between Britain and the United (<i>B-LOC, B-LOC</i>) States (<i>I-LOC, I-LOC</i>) to liberalise trans-Atlantic air traffic, which would create greater competition on the routes.
Harmful training instance: The man drew attention to himself in the North (<i>B-PER</i>) Island (<i>I-PER</i>) town of Tauranga while trying to reverse his car out of a pothole on Saturday night.

Table 8.4: Three selected test entities in the CoNLL 2003 NER dataset and corresponding most harmful training entities identified by the influence function. For each test entity, we report both its predicted and ‘true’ label (\hat{y} , y). The first two examples show a test entity tagged incorrectly by the model while the last example shows a test entity tagged correctly. For readability, we have normalized casing.

For the LSTM, we set the character embedding dimensionality to 100, and initialize weights via Xavier initialization (Glorot and Bengio, 2010). We set the dimensionality of the hidden layers to 300 and 100, for the word and character-level LSTMs, respectively. Tag predictions are made via a softmax layer directly connected to the hidden state vectors.⁴ This model achieves $\sim 88\%$ chunk level F1 on the CoNLL test set, and 58% chunk level F1 on Twitter test set.

Koh and Liang (2017) considered only unstructured (i.e., classification) settings, but many tasks in NLP are structured. We now describe ways to calculate the influence function in such cases.

Denote the i th training sentence by z_i , and the j th token in the sentence by $z_{i,j}$. Similarly, denote by $z_{\text{test},t}$ the t th token in the test sentence. To explain the predicted tags for a given test sentence, we consider two natural approaches: (1) treat the total loss on the complete test sentence as a whole; and (2) calculate influence w.r.t. individual constituent predicted entity spans. For (1), the influence

⁴We opt not to connect a Conditional Random Field (CRF) layer (Lample et al., 2016) for the sake of simplicity.

of training sentence z_i on z_{test} , $\mathcal{I}(z_i, z_{\text{test}})$, is:

$$\nabla_{\theta} \sum_t \mathcal{L}(z_{\text{test},t}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_{\theta} \sum_j \mathcal{L}(z_{i,j}, \hat{\theta}) \quad (8.2)$$

For (2), we consider each tagged entity in the test sentence independently; this affords insight into what influenced specific tag predictions. For each tagged entity, we calculate the influence function and retrieve the most influential annotated named entities (rather entire sequences) in the training set. Specifically, we denote a predicted entity $z_{\text{test},T}$ in the test sentence spanning a set of token position indices T , and an entity in a training sentence defined by token position indices J . Then, the influence function of the training entity w.r.t. the test entity, $I(z_{i,J}, z_{\text{test},T})$, is

$$\nabla_{\theta} \sum_{t \in T} \mathcal{L}(z_{\text{test},t}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_{\theta} \sum_{j \in J} \mathcal{L}(z_{i,j}, \hat{\theta}) \quad (8.3)$$

This effectively treats each entity within a sentence as a separate instance. In practice, we also include tokens immediately adjacent to a given entity to incorporate transition factors. We believe Eq. 8.3 is more intuitive than Eq. 8.2 and report results using this variant.

Harmful training entities identified via the influence function may be problematic or mislabeled. We present a few illustrative examples from the CoNLL corpus in Table 8.4, pairing each test entity with the training entity inferred to be most responsible for the loss corresponding to these predictions. Consider the harmful training example corresponding to the first example in the table: here ‘Millns’ is labeled as a *MISC* entity. However, it clearly refers to a person (*PER*). The influence function has identified this error as being responsible for the mislabeling of ‘Sung-Yeol’ as *MISC* (rather than the correct *PER*).

The second row in Table 8.4 shows a similar case. Here ‘Eurotunnel’ is erroneously labeled in the CoNLL dataset as a location (*LOC*), but actually refers to an organization (*ORG*). This error is responsible for the analogous mislabeling of ‘Zywiec’ (*PER*) in the test set as a *LOC*. Finally, in the last example, ‘United States’ is correctly tagged by the model (as a *LOC*). However, even for correct predictions,

the influence function still facilitates inspection of harmful training instances, i.e., training examples that nudged the model away from the correct prediction. Again this identifies a mislabeled entity in the CoNLL set: ‘North Island’, a *LOC* is incorrectly labeled as a *PER*.

The preceding examples suggest that by identifying training points responsible for misclassifications, one may efficiently improve the quality of annotated data. We now capitalize on this intuition by proposing a method for active transfer learning.

8.4 Active Transfer Learning via the Influence Function

In this section, we consider an active transfer setting in which we aim to train a model in a target domain \mathcal{T} , given: (1) access to a related, labeled source corpus \mathcal{S} ; and (2) a limited budget with which to acquire annotations in \mathcal{T} . The aim is then to exploit \mathcal{S} to maximize performance on \mathcal{T} under the budgetary constraints. The influence function provides a natural mechanism for selecting target points to annotate. Specifically: we can pseudo-label points in \mathcal{T} , train a model on these pseudo-annotations, and then re-label \mathcal{S} . The labels for the latter are known, and so we can calculate the induced loss; the influence function then facilitates identification of the most ‘harmful’ points in \mathcal{T} , w.r.t. \mathcal{S} ; these are prime targets for annotation.

More formally, suppose we have access to a source dataset $\mathcal{S} = \{z_0^s, z_1^s, \dots, z_{|\mathcal{S}|}^s\}$, where z_i^s denotes an instance/label tuple (x_i^s, y_i^s) . Now consider a target corpus $\mathcal{T} = \{z_0^t, z_1^t, \dots, z_{|\mathcal{T}|}^t\}$; to begin, we assume we have access only to instances x_i^t , no y_i^t , but have a modest budget to collect annotations in \mathcal{T} . How should we spend it?

Define a model f parameterized by θ . We first fit this on the source domain: $\hat{\theta}^S \leftarrow \arg \min_{\theta} \sum_{z_i^s \in \mathcal{S}} \mathcal{L}(f(x_i^s | \theta), y_i^s)$ producing a trained model f_{θ^S} . Next we ‘pseudo annotate’ all target instances with f_{θ^S} , yielding $\hat{\mathcal{T}} = \{\hat{z}_0^t, \hat{z}_1^t, \dots, \hat{z}_{|\mathcal{T}|}^t\}$, where $\hat{z}_i^t = (x_i^t, \hat{y}_i^t)$ and \hat{y}_i^t is the prediction from f_{θ^S} . The quality of these pseudo-annotations will depend on the correspondence between \mathcal{S} and \mathcal{T} . The idea is to exploit the available labels in \mathcal{S} using the influence function.

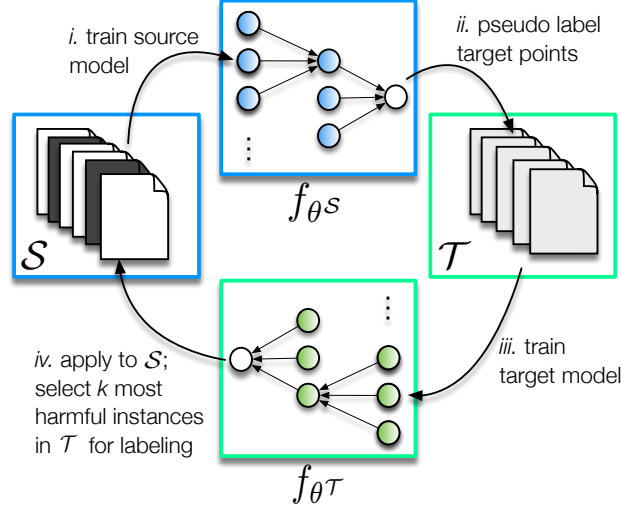


Figure 8.2: Active transfer via the influence function. Labels in \mathcal{S} are known; those in \mathcal{T} are not.

Concretely, we next use the induced pseudo annotations to train a model in \mathcal{T} : $\hat{\theta}^T \leftarrow \arg \min_{\theta} \sum_{z_i^t \in \hat{\mathcal{T}}} \mathcal{L}(f(x_i^t | \theta), \hat{y}_i^t)$. We will denote this model by f_{θ^T} . This has distilled the signal gleaned from the source corpus, as applied to the target corpus. Next we apply f_{θ^T} to all instances in the *source* dataset, yielding predictions $\{\hat{y}_0^s, \hat{y}_1^s, \dots, \hat{y}_{|\mathcal{S}|}^s\}$. Denote the source training set with the prediction from f_{θ^T} as $\hat{\mathcal{S}} = \{\hat{z}_0^s, \hat{z}_1^s, \dots, \hat{z}_{|\mathcal{S}|}^s\}$, where each \hat{z}_i^s is a tuple (x_i^s, \hat{y}_i^s) . For each we can calculate the loss under the true (known) label. For source instance i , the loss realized can be expressed as: $\mathcal{L}(\hat{y}_i^s, y_i^s)$. Large losses imply that the pseudo annotations for the responsible training instances were incorrect, i.e., that the transfer process failed in this case. Actively correcting these should afford rapid identification of what can and what cannot be transferred from \mathcal{S} to \mathcal{T} . This motivates the use of influence functions to identify the influential pseudo target points w.r.t the pseudo source points. Concretely, we calculate the total loss over \mathcal{S} under f_{θ^T} :

$$\mathcal{L}_s(\hat{\theta}^T) = \sum_{i=0}^{|\mathcal{S}|} \mathcal{L}(\hat{y}_i^s, y_i^s, \hat{\theta}^T) \quad (8.4)$$

Then we use Eq. 8.1 to calculate the influence of each target point (assuming its

pseudo label) on the loss incurred over \mathcal{S} as follows.

$$I(\hat{z}_i^t, \hat{\mathcal{S}}) = \nabla_{\theta} \mathcal{L}(\hat{z}_i^t, \hat{\theta}^T) H_{\hat{\theta}^T}^{-1} \nabla_{\theta} \mathcal{L}_s(\hat{\theta}^T) \quad (8.5)$$

The above process applies to the simple binary classification task. For sequence tagging tasks such as NER, we consider each individual entity as an instance as Eq. 8.3. Then for each of named entity in the pseudo target set, we calculate the influence function:

$$I(\hat{z}_{i,j}^t, \hat{\mathcal{S}}) = \nabla_{\theta^T} L(\hat{z}_{i,j}^t, \hat{\theta}^T) H_{\hat{\theta}^T}^{-1} \nabla_{\theta^T} L_s(\hat{\theta}^T) \quad (8.6)$$

where $\hat{z}_{i,j}^t$ denotes the j th pseudo annotated entity in the i th target training sentence, and $L(\hat{z}_{i,j}^t, \hat{\theta}^T)$ is the total loss of all the tokens in the j th entity. The smaller $I(\hat{z}_i^t, \hat{\mathcal{S}})$ is, the more harmful the pseudo annotated target point is to the source point, indicating domain mismatch. Our strategy is thus to select the most harmful points for annotation. In the sequence tagging case, this is a bit more complicated, because loss is calculated w.r.t. source and (pseudo-labeled) target *entities*.

Note that this strategy is not necessarily restricted to transfer learning situation. But data points from two different domains tend to have more mismatch than points from the same points. Since influence function can help find domain mismatch, it is more suitable for the domain transfer scenario than the generic active learning.

Algorithm 2 Active Transfer Learning via the Influence Function

- 1: Train model f_{θ^S} on source domain data \mathcal{S}
 - 2: Use f_{θ^S} to pseudo label target domain, obtaining pseudo target points $\hat{\mathcal{T}}$
 - 3: Train a model f_{θ^T} on $\hat{\mathcal{T}}$
 - 4: Use f_{θ^T} to predict labels on \mathcal{S} , obtaining predicted source set $\hat{\mathcal{S}}$
 - 5: Use Eq. 8.5 to calculate the influence of each pseudo labeled target point on $\hat{\mathcal{S}}$
 - 6: Pick the k most harmful pseudo labeled target points and annotate them
 - 7: Transfer learning: fine-tune f_{θ^S} on the k labeled target points =0
-

In practice, rather than picking one single instance, we identify the k most harmful points for annotation, where k reflects a budget. We then perform transfer

learning using the model trained on the source domain $f_{\theta s}$ and these k target points.

Our work follows the simple transfer learning strategy in (Mou et al., 2016): fine-tune $f_{\theta s}$ using the selected k target points. We summarize the proposed active sampling and transfer learning strategy in Figure 8.2 and Algorithm 2.

We note that there are three other obvious transfer learning setups. One could: (1) train on both the source and selected target points, initializing the model with $f_{\theta s}$; (2) train a randomly initialized model jointly on both source and available target points; or (3) define source and target networks that comprise both shared and independent parameters (Bousmalis et al., 2016; Daumé III, 2009; Yang et al., 2017), then train the common network on both the source and target data while training the shared network only on the respective data. This network is initialized randomly.

We explored all three of these alternative baselines but none achieved significant gains over simply fine-tuning $f_{\theta s}$ using target data. All variants outperform training the model on *only* target data after random initialization (i.e., transfer learning outperforms learning sans transfer). We also note that our focus here is not the transfer learning strategy, but rather the sampling strategy in transfer learning: how to collect better target points to label to expedite the transfer learning process. Our proposed sampling method can be applied to any transfer learning method.

8.5 Experiments

We conduct experiments on text classification (sentiment analysis), sequence tagging (NER), and multi-genre natural language inference tasks. For all, we assume that initially there are no target training points available, and that we have access to only a small annotation budget, sufficient to acquire k labeled instances in the target domain, collected according to Algorithm 2.

Baselines. We compare the proposed influence approach to a few alternative sampling methods (scoring function): 1) Random (i.i.d.) sampling; 2) Uncertainty sampling (i.e., entropy): selecting target points with the highest score, as evaluated under $f_{\theta s}$ (entropy); 3) (Rai et al., 2010)’s active transfer method using ‘domain

classifier’ entropy (DC-entropy); 4) least similar representation, we calculate the Euclidean distance between each pair of labeled instance and target unlabeled instance in the final representation layer, and then for each unlabeled instance, calculate the minimum distance across all labeled instances as the scoring function. The active learner should choose those least similar points (with maximum minimum distance to the labeled points). With uncertainty sampling in the NER task, we calculate the entropy as a sum of constituent token entropies. When using (Rai et al., 2010)’s method, we first train a ‘domain classifier’ (which discriminates between source and target points). Then, during sampling we apply the entropy strategy only on the target training points correctly classified as target points by the domain classifier (DC-entropy). In the sequence tagging task, the domain classifier is trained by feeding the last time step of LSTM into a softmax classifier. We note that there are other active transfer learning methods (Wu et al., 2017; Li et al., 2012) designed for certain type of linear classifiers, but can’t be directly applied to neural models.

In all sets of experiments below, we vary the simulated budget k from 50 instances to 20% of the total available target training set instances.

8.5.1 Active Transfer: Text Classification

For text classification, we perform multiple sets of experiments. First, we do document classification. We use source and target reviews from different domains (Blitzer et al., 2007) that have been traditionally investigated for transfer learning for sentiment analysis. In transfer learning, the model f_{θ_S} is the same LSTM architecture used in Section 8.3. The model trained on the pseudo-annotated target set, f_{θ_T} , is a bi-gram logistic regression model.

Figure 8.3 reports transfer results (learning curves) averaged over 5 runs for some pairs. Unfortunately, we find that for document classification, the active learning is hard to make progress in the transfer setting. Though the initialization point might be helpful compared to not using transfer learning, the learning curve is hardly increasing. We also find the domain classifier can achieve almost 100% accuracy, so it has the same performance as the simple entropy method. We won’t report their curve in the following results.

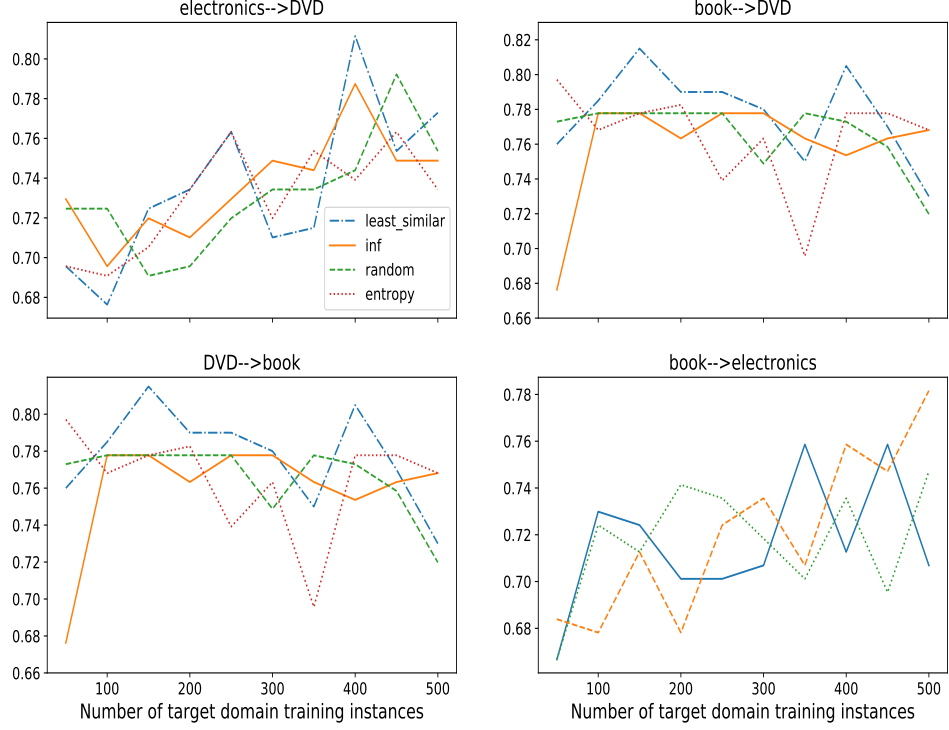


Figure 8.3: Target domain accuracy (y-axis) of transfer learning for document classification using LSTM with varying annotation budget (x-axis) for the influence function approach (‘Inf’) vs. 3 baselines.

Different than document classification, LSTM on short sentence classification in transfer learning setting works better. We use movie review, customer review and subjectivity detection dataset as the source or target domain. The results are shown in Figure 8.4. The corresponding AUC scores are shown in Table 8.5.

We also experiment with CNN for sentence classification using the above dataset. The results are shown in Figure 8.5, and the corresponding AUC scores are shown in Table 8.6.

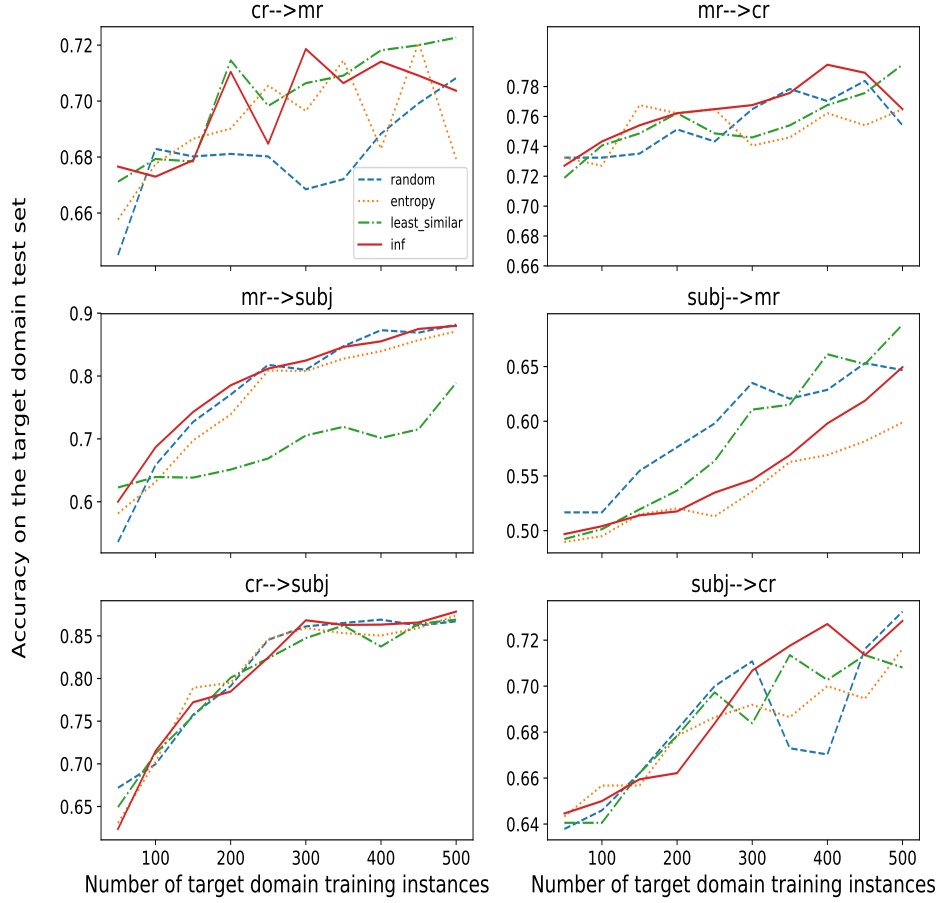


Figure 8.4: Target domain accuracy (y-axis) of transfer learning for short sentence classification using LSTM with varying annotation budget (x-axis) for the influence function approach (‘Inf’) vs. 3 baselines.

8.5.2 Active Transfer: Sequence Tagging

We conduct two sets of experiments on sequence tagging corpora: transfer from CoNLL to Twitter (‘C to T’) and vice versa (‘T to C’). For $f_{\theta S}$, we use the same LSTM model as described above (8.3.2); we also use this model as $f_{\theta T}$. As

	Random	Entropy	Least Similar	Influence
cr to mr	0.942	0.960	0.972	0.966
mr to cr	0.951	0.947	0.951	0.964
mr to subj	0.892	0.874	0.774	0.903
subj to mr	0.866	0.781	0.848	0.803
cr to subj	0.926	0.924	0.919	0.924
subj to cr	0.932	0.930	0.935	0.942

Table 8.5: Area Under learning Curves for sentence classification achieved using active transfer methods with LSTM (corresponding to Fig. 8.3).

	Random	Entropy	Least Similar	Influence
cr to mr	0.964	0.962	0.969	0.973
mr to cr	0.944	0.955	0.909	0.959
subj to cr	0.905	0.924	0.894	0.924
cr to subj	0.905	0.922	0.788	0.983
mr to subj	0.961	0.972	0.788	0.983
subj to mr	0.958	0.913	0.942	0.929

Table 8.6: Area under learning curves for sentence classification achieved using active transfer methods with CNN (corresponding to Fig. 8.5).

mentioned above, the Twitter dataset defines 10 types. For this work, we map these onto the 5 CoNLL corpus types as follows: *person* \mapsto *person*, *geo-location* \mapsto *location*, *company* \mapsto *organization*, *facility* \mapsto *misc*, *product* \mapsto *misc*, *music artist* \mapsto *person*, *movie* \mapsto *misc*, *sports team* \mapsto *organization*, *tv show* \mapsto *misc*, and *other* \mapsto *misc*.⁵

Figure 8.6 shows learning curves results averaged over 5 runs. Table 8.7 shows the (normalized) mean AUC of each method. The result is mixed: for CoNLL to Twitter, the influence function method is the best most of time, but for Twitter to CoNLL, the influence function is worse than entropy approach.

8.5.3 Active Transfer: Multi-Genre Natural Language Inference

We perform experiments on multi-genre natural language inference (MultiNLI) tasks, and use Multi-Genre NLI corpus (Williams et al., 2017). In this corpus, each

⁵The influence function requires the two domains to share the same label space.

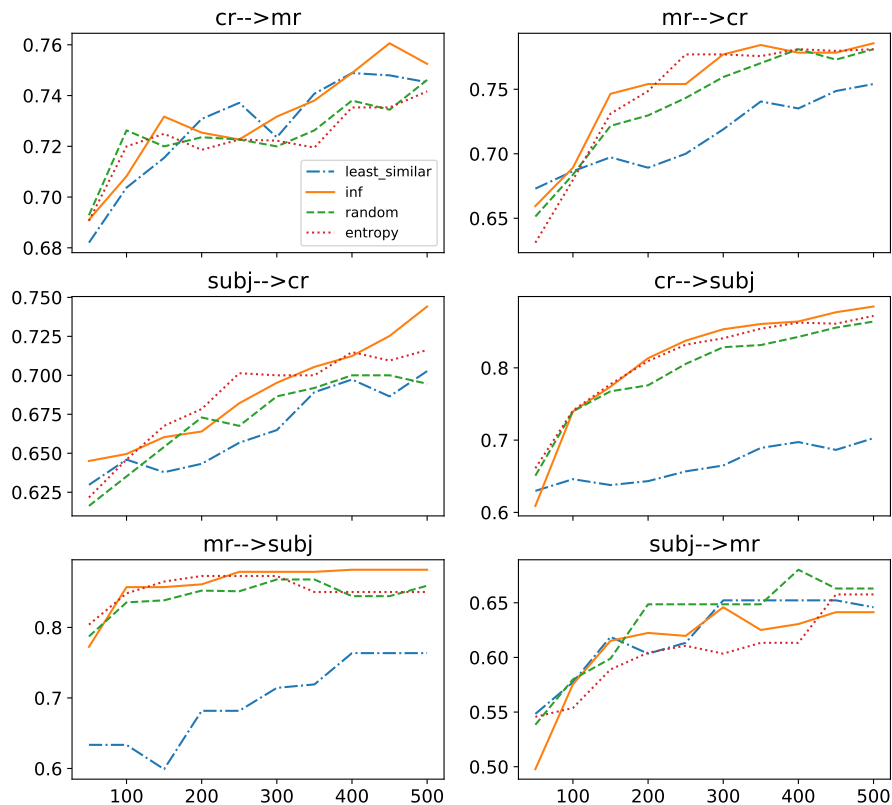


Figure 8.5: Target domain accuracy (y-axis) of active transfer learning for text classification (i.e., sentiment analysis) using CNN with varying annotation budget (x-axis) for the influence function approach (‘Inf’) vs. 3 baselines.

	Random	Entropy	Least Similar	Influence
C to T	0.827	0.850	0.842	0.873
T to C	0.960	0.986	0.817	0.977

Table 8.7: AUCs for sequence tagging (NER) achieved by different active transfer learning strategies (corresponding to Fig. 8.6). ‘C to T’ denotes transferring from CoNLL to Twitter, while ‘T to C’ denotes the reverse.

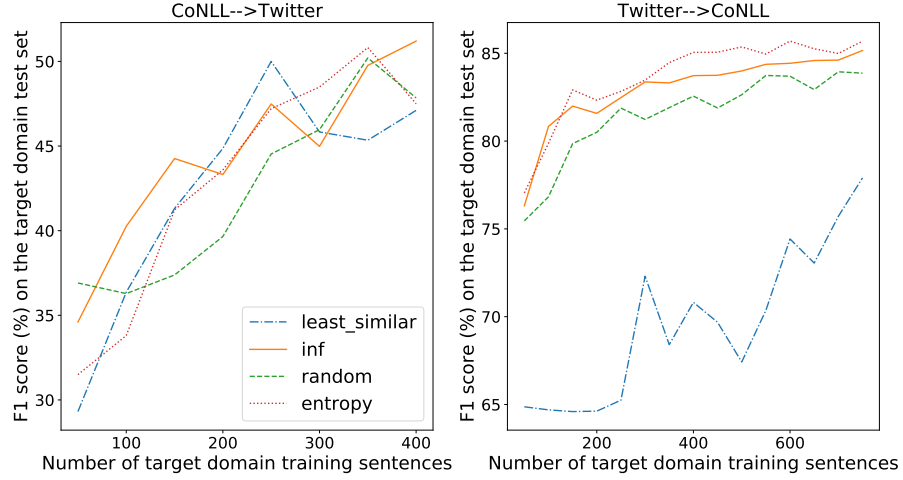


Figure 8.6: Target domain F1-score (y-axis) of transfer learning for sequence tagging (i.e., NER) with varying annotation budget (x-axis) for the influence function approach (‘Inf’) vs. the 3 baselines.

example in the consists of a pair of sentences, and the model should predict whether the relationship between the two sentences is *entailment*, *neutral* or *contradiction* as a three-class classification problem. The corpus contains five genres: *fiction*, *government*, *slate*, *telephone* and *travel*. We use CBOW model as described in (Williams et al., 2017).

We show some experimental results using some pairs in Figure 8.7. The corresponding AUC scores for the learning curve are shown in Table 8.8. As we can see from the learning curves, there is a large variance in the performance. Although the starting point is better than random initialization, the performance sometimes drops down a lot after some increase. For example, in ‘government to travel’, when there are 3,000 target labels, the accuracy of the influence function method can achieve about 52.5%, but when there are 4,000 target points, the accuracy decreases to about 45.5%, even lower than the beginning. Although the influence function method does the best most of time from Table 8.8 in terms of the AUC score, their learning curves are quite mixed compared with other methods from Figure 8.7. For example, in ‘fiction to travel’, the influence function method is the worst among all

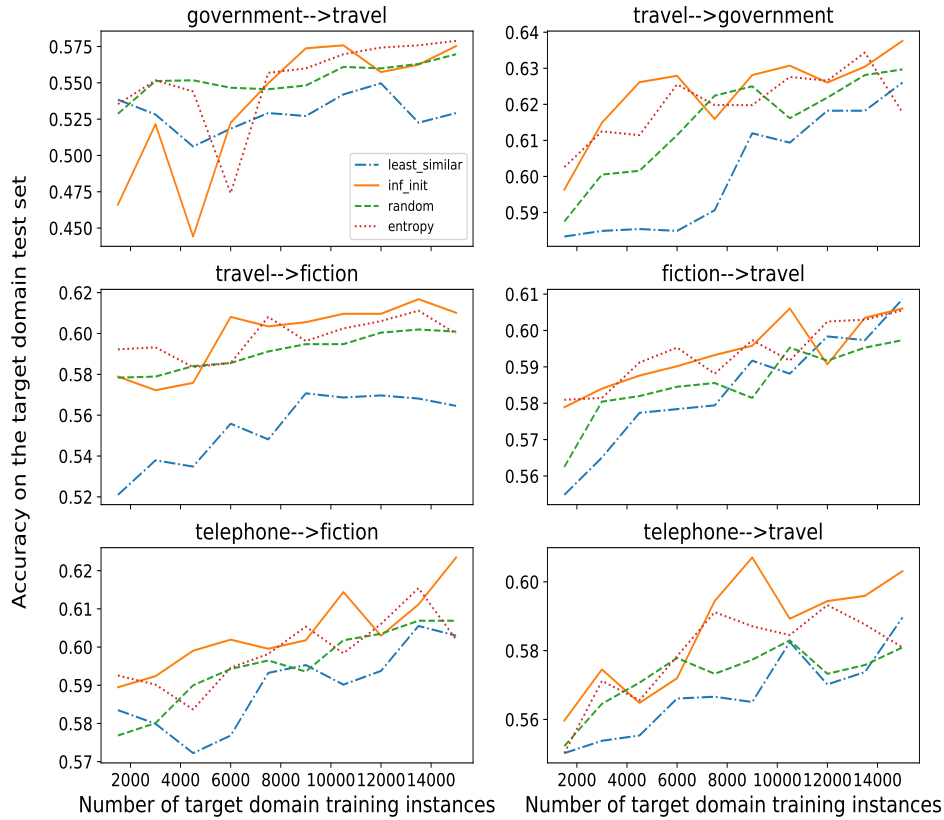


Figure 8.7: Active transfer learning curve for MultiNLI. y-axis is target domain F1-score and x-axis is the number of labeled target instances.

the methods when the number of target instances reaches 12,000.

8.5.4 Failure analysis and possible improvement

From the above figures and tables, we can see that influence function based active transfer learning outperforms strong baselines in some cases, but this result is not consistent. In some other cases it does worse than entropy and random sampling, such as when transferring subjectivity to movie review in Figure 8.4. We

	Random	Entropy	Least Similar	Influence
G to T	0.955	0.953	0.913	0.927
T to G	0.965	0.974	0.942	0.979
T to F	0.970	0.980	0.910	0.983
F to T	0.963	0.976	0.960	0.976
TE to F	0.978	0.984	0.968	0.991
TE to T	0.942	0.954	0.932	0.963

Table 8.8: AUCs for MultiNLI achieved by different active transfer learning strategies (corresponding to Fig. 8.7). ‘G’ is government. ‘T’ is travel. ‘F’ is fiction. ‘TE’ is telephone

also observe large variance in most learning curves. This might be caused by initialization using a good source model. Though the source model can provide a better starting point in the learning curve, it is harder for the target model to make fast and consistent progress based on that. One explanation is that the model gets trapped in a bad local area, and fine-tuning fails to help it escape away from that local area. This might motivate using a better initialization method. One possible solution would be using a source model less fitted to the source data, so that it is easier for the fine-tuning process to push the model away from the local optimum of the source model.

One limitation of our method is its problematic use of the influence function. The assumption of the influence function only holds when the loss function is convex (Koh and Liang, 2017). When the loss is not convex, such as in the neural networks case, it needs to be calculated approximatedly. In our case, we first train the whole neural networks and then freeze the layers before the softmax layer and then fine-tune the softmax layer. In this way, the neural layers can be viewed as a feature extractor for the softmax layer which is itself convex. When calculating the influence function, we only use softmax layer parameters. But this process is only approximating what the influence function truly wants to discover. The misaligned target points identified by the influence function in this way might not be the exact ones that are most harmful to the target model. It is worth studying how to more accurately identify responsible training points using influence function for

neural models. One possible alternative way of calculating the influence function is that after further fine-tuning softmax layer parameters on top of frozen lower layers, including all parameters of the neural model into calculation of the influence function. Another possible way of overcoming the non-convex issue is modifying the Hessian in Equation 8.1. For example, one could add some damping term to the Hessian matrix to make it positive definite to avoid the non-convex problem.

Another drawback of our method is that it is estimating the mismatch between each target point and the whole source set in Equation 8.1. Using all the source points to calculate the influence function would easily result in an average effect when selecting the harmful target points. An alternative option is to only consider those source points that have higher loss calculated from the target model. For example, we could set a threshold on the loss value and remove source points that have lower loss. In other words, we first pick misaligned source instances with the target, and then for these source instances we pick corresponding harmful target points. This might result in more misaligned target points.

Currently, we use two independent models for the source and the target. That is, the source model is exclusively used for transferring to the target, while the target model is only used for finding harmful target points. This might be problematic because which target points are maximally misaligned with the source largely depend on how the target model performs. But the target model's performance is loosely connected to the source model. It might be worth trying to more closely couple these two models. For example, the two models could share part of the parameters to more directly influence each other.

Specifically for NER models, we can see from Figure 8.6 that the influence function does not help compared to the entropy method. When calculating the influence function for NER models, we currently only consider the loss on named entities, but ignore the loss on the other tokens. This might be a problem since all the words in the sequence contribute to the loss and in turn affect the influence function value. So it might be worth trying incorporating all the tokens when calculating the influence function or considering the length of each sequence. Another problem of our method is that the target sequence model is an LSTM which again is non-

convex. It might be worth trying some other convex function such as conditional random field as the target model. Or similarly as in the general text classification case, after training LSTMs, keep fine-tuning the top softmax layer while freezing lower layers. In this way, the LSTM layers are feature extractors and the softmax classifier is a convex function.

8.6 Chapter Summary

In this chapter, we have proposed using the recently derived *influence function* to debug and transfer neural NLP models. We qualitatively demonstrated that the influence function generalizes to language tasks. We then modified the influence function to accommodate sequence tagging, specifically by deriving a variant that provides the influence of specific train spans (e.g., entities) on a given test span. We showed that one can use this approach to identify problematic/mislabeled instances in an NER task.

Next, we propose a novel active transfer learning strategy for text classification, sequence tagging and multiNLI that uses the influence function to identify target points that seem misaligned with the source corpus. However, from the experimental results, we can see that this method sometimes does worse than other baseline approaches. Overall, we think that this method is not applicable given its overall mixed performance. This is possibly caused by the approximatedly calculated influence function or the estimation of the mismatch between the source and the target. We claim that more future research on this method is necessary to get it to work.

Chapter 9

Active Learning with Pre-trained BERT

9.1 Chapter Overview

In the last chapter, we discuss active transfer learning where we initialize the model with a pre-trained model from a source domain and perform active learning based on that¹. But another more prevailing transfer learning method is using pre-trained language models. Pre-trained language models (LMs) such as the Bidirectional Encoder Representations from Transformer (BERT) have recently afforded impressive gains across a variety of NLP tasks. Such models have the potential to allow rapid adaptation to new domains, as one needs only to fine-tune parameters, rather than learning them from scratch. In this chapter we set out to investigate whether we can further economize annotation effort via AL. Does combining AL with LM pre-training offer efficiency gains over using either strategy alone? We explore standard AL heuristics and strategies specifically designed for adapting BERT to text classification and sequence tagging tasks. We find that AL affords rapid adaptation of deep pre-trained LMs, and that using AL heuristics designed for deep LMs tends to offer the strongest gains.

9.2 Prior Work

For prior work on active learning, we refer the reader to Chapter 7. This section mainly reviews prior works on pre-training deep neural NLP models.

In more recent efforts, researchers in NLP have begun to seriously investigate unsupervised pre-training methods, thus exploiting an effectively unlimited volume of available data (i.e., text). This can be used to train a deep, highly parameterized language model, which can then be fine-tuned with whatever data is available for a particular target task. For instance, ELMo (Peters et al., 2018) com-

¹This chapter's work is unpublished.

bines token embeddings with L layer bi-directional LSTM output vectors in a single vector as a weighted sum, and learns the linear weight during the fine-tuning on a target task. In this way, ELMo can capture contextual information of words.

Howard and Ruder (2018b) proposed a Universal Language Model Fine-tuning (ULMFiT). They first pre-train an LSTM-based language model. They then fine-tune this model on target data using a discriminative fine-tuning procedure that treats different layers of LSTMs differently during estimation, in particularly by associating individual learning rates with different layers. Their approach then gradually unfreezes each layer during fine-tuning of the model for the target task.

Radford et al. uses a transformer decoder (Vaswani et al., 2017; Liu et al., 2018) in place of LSTM to train a language model. Transformer is a multi-layer self-attention network that captures the long range interaction between each pair of tokens in the sentence (Vaswani et al., 2017). Later during fine-tuning, they add task-specific input transformations for each task, and feed that into the pre-trained transformer. Finally, they add a linear softmax layer on top of the transformer output. Devlin et al. (2018) also uses transformer to do the pre-training, but they propose a novel unsupervised training. They randomly mask 15% input tokens in the sentence, and predict these masked tokens as the target using transformer. They also introduce a next sentence prediction task by predicting whether the second sentence is the next sentence of the first sentence. All of these works aim to pre-train models that yield good representations for arbitrary downstream tasks, so that in turn it will be easier to adapt light-weight models that consume these (e.g., a single dense layer with a softmax for classification). We will use this property of the pre-trained language model in our active learning approach.

9.3 AL with BERT (Active BERT)

9.3.1 Preliminaries on BERT

BERT (Devlin et al., 2018) uses a multi-layer transformer (Vaswani et al., 2017) encoder to pre-train a masked language model. During pre-training, $\sim 15\%$ of all wordpiece tokens (Wu et al., 2016a) in the sentence are randomly masked.

The transformer attempts to reconstruct the entire input sequence, i.e., predict the masked tokens via a linear layer with parameters W_{LM} . BERT is additionally trained via a second objective related to predicting the next sentence.

Fine-tuning BERT for classification simply entails adding a task-specific softmax layer on top of the transformer output, and fine-tuning all parameters. Note that in the fine-tuning process, the parameters W_{LM} for predicting masked tokens are ignored. However, below we propose a scoring function that will use the complete pre-trained model (including W_{LM}) as a means to measure the quality of target instance representations.

9.3.2 BERT for Active Learning

Our first aim in this work is to explore whether active learning e.g., uncertainty sampling is useful for BERT. This is worth investigating because when the target training data is limited, it is possible that BERT is overfit on the large unsupervised corpus and hard to transfer on the target data. So it is interesting to see how the model performs when there is small target training data and whether active learning is helpful compared to the naive random sampling.

In addition to exploring the aforementioned scoring function, we propose a new method that tries to utilize the property of BERT. BERT can first extract a representation for the text instance through the pre-trained transformer network. This representation can be viewed as a feature representation, which is then fed into a task-specific softmax layer. If the feature representation is good, then the softmax layer can also be learned easily. Otherwise, it is difficult for the softmax layer to adapt itself on top of bad features. So in AL, we tend to choose those text instances already with good representation, so that the learner can quickly train a better softmax layer.

To evaluate the representation of the instance (how good the feature represent is), we follow BERT’s pre-training step (Devlin et al., 2018). For each unlabeled instance x_i in \mathcal{U} , we randomly mask approximately 15% input tokens, and calculate the probability of these masked token as output using the pre-trained transformer. The masked perplexity for the instance is the average masked token loss,

denoted by $\phi_{\text{masked perplexity}}$. During active learning, we tend to choose those points with lower perplexity indicating that the transformer already extracts a good representation for the softmax layer. Note that in this instance selection phase, the projection layer W_{LM} in masked language model is used to evaluate the probability of each masked token. In the training phase, it is still discarded, and only the transformer and softmax parameters get trained. So after the transformer gets fine-tuned in the first iteration of AL, W_{LM} (not trained during fine-tuning) will be inconsistent with the transformer. We thus can't totally rely on it. So the scoring function is defined as an interpolation between entropy and perplexity.

$$\phi = \phi_{\text{entropy}} + \lambda(1 - \phi_{\text{masked perplexity}}) \quad (9.1)$$

To make entropy and masked perplexity comparable, we normalize entropy and perplexity to the range between 0 and 1 by dividing by their respective maximum value in the current iteration. λ is set to the reverse of the number of AL iterations, such that as the learning goes, the perplexity will have smaller weight. Since we want to choose instances with lower masked perplexity, we minus it from 1.

Note that one alternative thought might be choosing instances with bad representation through the transformer to let the learner improve on these instances. However, we find that it is hard for the large pre-trained transformer to get improved with limited training data. So instead, we pick instances that already has good representations in order to improve the softmax layer.

9.4 Experiments

We perform experiments on standard text classification and sequence labeling (NER) tasks. For the former, we use the following corpora: Movie reviews (MR) (Pang and Lee, 2005b), customer reviews (CR) (Hu and Liu, 2004), The Corpus of Linguistic Acceptability (COLA) (Warstadt et al., 2018), Subjectivity Detection (Subj) (Pang and Lee, 2004), question classification (TREC) (Li and Roth, 2002), and opinion mining (MPQA). For NER, we use the CoNLL named entity recognition (NER) task (Tjong Kim Sang and De Meulder, 2003), a Twitter

	#train	#eval
MR	9603	1059
CR	3388	387
TREC	5452	500
Subj	8979	1021
COLA	8551	1043
MPQA	9586	1020
CoNLL	14987	3466
Twitter	1915	999
EBM-NLP	16785	639

Table 9.1: Statistics of the classification (top) and sequence tagging (bottom) corpora used for experiments. #train denotes the initial total number of unlabeled instances in the pool, #eval the number we evaluate on.

NER task (Ritter et al., 2011), and an evidence based medicine (EBM) tagging task, EBM-NLP (Nye et al., 2018).

We point the reader to the respective sources for detailed descriptions of the data and tasks, but provide basic corpora statistics in Table 9.1. For datasets without a pre-defined train/eval split, we created one. We set the labeling budget to $\sim 20\%$ of the available training data. We use pre-trained BERT as initialization, and for EBM-NLP we use the domain specific Bio-BERT (Lee et al., 2019).

We compare the following AL strategies: (1) random (i.i.d.) sampling; (2) uncertainty sampling via entropy;² (3) EGL-wordpiece, i.e., applying EGL method in Chapter 7 on the subword embedding space (4) (lowest) perplexity; and (5) entropy + perplexity, i.e., the interpolation just described.³ We initialize to the pre-trained BERT small (Devlin et al., 2018) in all cases save for EBM-NLP, for which we also explore use of the domain-specific Bio-BERT (Lee et al., 2019).

²For sequence tagging tasks, we use the total token entropies over the sequence as the scoring function.

³We also considered length-normalized least confidence method (Shen et al., 2017), but it performed worse than entropy in preliminary experiments.

9.5 Results

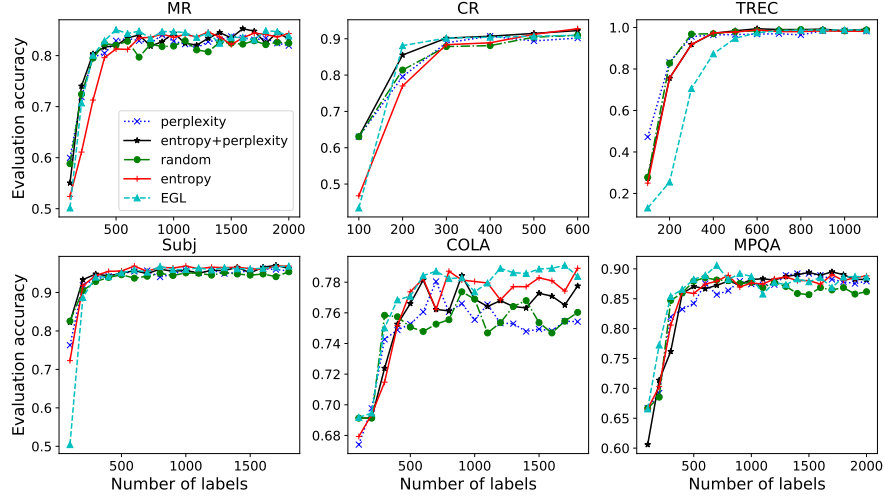


Figure 9.1: Active Learning Curve for classification dataset.

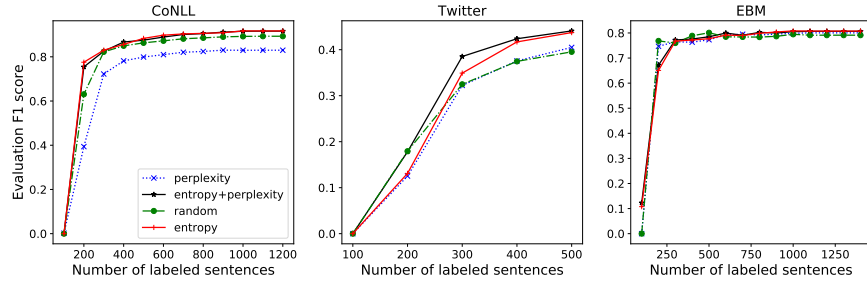


Figure 9.2: Active Learning Curve for NER dataset.

Learning curves averaged over five runs for these methods are shown in Figure 9.1 for text classification tasks and in Figure 9.2 for NER tasks. and AUC scores for all corpora are reported in Table 9.2.

Table 9.2 summarizes all results via the corresponding average area under the curve (AUC) measures. We observe that some variant of AL outperforms

	ran	ent	per	ent + per	EGL
MR	96.1	95.3	96.7	96.6	97.0
CR	91.6	89.5	91.7	93.9	92.0
TREC	93.0	92.4	93.8	94.2	82.2
Subj	96.1	97.2	96.4	97.3	95.3
COLA	93.6	94.6	93.7	94.2	95.5
MPQA	95.4	95.5	95.4	95.9	96.0
CoNLL	88.5	91.7	79.9	91.4	-
Twitter	61.0	63.2	58.2	68.5	-
EBM-V	91.1	88.6	87.8	88.3	-
EBM-B	93.6	93.7	93.6	94.2	-

Table 9.2: AUC scores averaged over five runs for each method on each dataset. ‘ran’ is random sampling, ‘ent’ is entropy, ‘per’ is perplexity, ‘ent+per’ is entropy + perplexity. Corpora above the horizontal line are text classification tasks; below are sequence tagging. ‘EBM-B’ denotes Bio-BERT initialization on EBM, and ‘EBM-V’ denotes the standard version.

i.i.d. random sampling on all but one dataset (more on this case below). Furthermore, models specifically designed for adapting neural LMs (namely ent + per and EGL) outperform standard entropy-based uncertainty sampling in all but one case (CoNLL; a difference of 0.03 F1).

For EBM-NLP (Nye et al., 2018), we report results using both Bio-BERT (EBM-B) and ‘vanilla’ BERT (EBM-V) for initialization. Interestingly, when the ‘out of domain’ initialization (EBM-B) is used, AL performs *worse* than i.i.d. sampling. But when we use in-domain initialization, AL outperforms i.i.d., using the proposed method. (Figure 9.1 shows results using BioBERT only.) Note that we are not simply making the (obvious) observation that pre-training in-domain is generally helpful: this boosts the performance of *both* AL and random sampling, but the comparative performance flips.

9.6 Comparison with AL methods in Chapter 7 and Chapter 8

In Chapter 7, Chapter 8 and Chapter 9, we develop three AL strategies. In this section, we compare their different usages. In Chapter 7, we develop the EGL

approach for CNN. This is suitable for any general text classification tasks. In Chapter 8, we develop an influence function based AL method. But that method is more suitable to the transfer learning scenario, in which we can pre-train a model on a source domain, but our target is to adapt this model to the target domain. At each round of AL, the goal is select target points that mostly mismatch with source points. In this chapter, the AL method is based on a pre-trained language model which requires perplexity as the scoring function. The major difference between Chapter 8 and this chapter is that in Chapter 8, the source domain needs to be the same task as the target domain, but in this chapter, the source domain pre-trains a language model regardless of the target domain.

9.7 Chapter Summary

We observe that *active learning methods offer efficiency gains even when using BERT*,⁴ suggesting that AL combined with deep pre-trained language models is a promising strategy for learning in low supervision settings. We additionally observe that *AL methods bespoke for neural architectures (namely ent+per and the EGL over wordpiece methods) tend to perform better than traditional AL approaches*.

⁴Performance without using BERT, i.e., training from scratch, is consistently worse, as one would expect; we do not report these results here.

Chapter 10

Conclusion and future work

Neural models have been shown to work well when there is large amount of training data. However, we do not often have large scale training data because annotating data is expensive. There are many strategies proposed for improving model performance in such ‘low-supervision’ scenarios. In this thesis we investigate three strategies: training neural models beyond instance labels, transfer learning for multiple types of source data, and active learning for neural models.

We first consider incorporating other types of resources into neural model training beyond instance labels. In Chapter 3, we investigate the scenario where we not only ask human annotators to label the training examples, but also to provide further rationales that support their annotations. These rationales can augment the limited training labels, since the model can be trained on both the instance labels and the rationale labels. Then in Chapter 4, we consider how to incorporate ontologies as a prior knowledge into neural models. We propose a novel training method that forces similar words in semantic lexicons to share more similar weights in the embedding space to inductively bias the neural model training.

In addition to the above described methods that use other types of resources, one of the mostly commonly used methods for low-supervision problems is transfer learning. When there is not enough data for the target domain we are interested in, we can often use large available training data from another source domain. Typically, we can first train a model on the source domain, and then transfer or adapt the trained model to the target domain. Previous works have studied general transfer learning where they transfer a single source model to a single target model. However, it is often the case that the source domain contains multiple types of data, and simply mixing them together might not be the best option. Instead, we study how to better leverage various available source data to better suit the source to the target domain. In Chapter 5, we study how to better leverage multiple sets of pre-trained word embeddings trained from different domains in neural models and fine-tune

them intelligently on the target domain. In Chapter 6, we study domain adaptation for text generation. We develop a novel encoder-decoder model that uses various writing styles (domains) in the training data and performs domain adaptation at generation time. The model is able to generate text with any specific style even if the style has little training data or is totally unseen in the training set.

Then we consider the most straightforward way to deal with low-supervision situations: collecting more training data but under a limited budget. We study how to collect more valuable labels if we are not allowed to annotate much data. In Chapter 7, we develop new active learning (AL) methods to collect more informative examples to be annotated specifically for neural models, so that better models and more discriminative text representation can be learned with fewer labels. Following this, we consider a situation where we not only collect a limited amount of labels from the target domain using AL, but at the same time, we also pre-train a good model from a richly annotated relevant source domain and transfer it to the target domain. We investigate this issue of combining active learning and transfer learning in Chapter 8. Other than transferring from a relevant supervised source domain, in Chapter 9 we also study AL strategies on top of a pre-trained deep language model. Specifically, we investigate how to actively select target instances using pre-trained transformer encoders.

In short, this thesis has considered three strategies for low-supervision scenarios: training neural models beyond instance labels, transfer learning and active learning specifically designed for neural models. We propose novel methods for each of the three strategies and improve model performance.

10.1 Future Work

10.1.1 Rationales in Active Learning

In Chapter 3, we introduce a novel neural model that can incorporate rationale annotations into training. This model can not only have stronger predictive accuracy, but can also provide better explanations at test time for its prediction. This makes the model interpretable. In Chapter 7 and Chapter 9, we develop novel active

learning methods specifically designed for neural models. One future direction is to combine these two goals more closely, that is, asking human annotators to annotate rationales in the active learning loop. In this way, rationale augmented neural models can further expedite the active learning. And more rapid active learning can in turn generate better explanations. It is worth investigating the scoring function for instances and rationales and also whether to annotate rationales or instance labels in the current iteration.

10.1.2 Rationales for Structured Neural Models

In Chapter 7, we use rationales to augment text classification models. But so far, there is no prior work using rationales to augment structured models, for example, sequence models for named entity recognition (NER). One future direction would be asking human annotators to mark rationales for NER datasets. For each word, the annotator should mark which surrounding words can explain this word's entity tag. Then similar to RA-CNN in Chapter 7, the sequence model should put more weight on those rationale words than other words in the sentence.

10.1.3 Active Learning for Structured Neural Models

In Chapter 7, we develop the EGL-word method aiming at improving embedding layer for the downstream tasks. The intuition is that in feed-forward networks, a better and a more discriminative feature representation layer can facilitate higher layer learning. However, this does not necessarily hold for structure models. For example, in an RNN, the current output depends on lower layers' parameters, but also depends on previous steps' outputs. So in order to improve each step's output, we should not only consider the expected change on lower layers' parameters, but also take into account previous steps' output. This is more complicated than feed-forward networks.

References

- David Alvarez-Melis and Tommi Jaakkola. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 412–421, 2017.
- Nabiha Asghar, Pascal Poupart, Xin Jiang, and Hang Li. Deep active learning for dialogue generation. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 78–83, 01 2017.
- Ramon F Astudillo, Silvio Amir, Wang Lin, Mário Silva, and Isabel Trancoso. Learning word representations from scarce and noisy data with embedding subspaces. In *Proceedings of Association for Computational Linguistics*, pages 1074–1084, 2015.
- Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226, 2009.
- Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.
- John Blitzer, Mark Dredze, Fernando Pereira, et al. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.
- Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

- Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010.
- Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.
- Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.
- Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proceedings of Association for Computational Linguistics*, pages 136–145, 2012.
- Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 740–750, 2014.
- Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *ICML*, pages 2285–2294, 2015.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.
- Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML workshop on challenges in representation learning*, volume 2, 2013.
- Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*, pages 93–98, 2016.
- Mei Chung, Denish Moorthy, Nira Hadar, Priyanka Salvi, Ramon C Iovin, and Joseph Lau. *Biomarkers for Assessing and Managing Iron Deficiency Anemia in Late-Stage Chronic Kidney Disease*. AHRQ Comparative Effectiveness Reviews. Agency for Healthcare Research and Quality (US), Rockville (MD), 2012.

- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- Aaron M Cohen, William R Hersh, K Peterson, and Po-Yin Yen. Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, 13(2):206–219, 2006.
- Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine learning*, pages 160–167, 2008.
- Issa J Dahabreh, Denish Moorthy, Jenny L Lamont, Minghua L Chen, David M Kent, and Joseph Lau. Testing of cyp2c19 variants and platelet reactivity for guiding antiplatelet treatment. 2013.
- Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- Finale Doshi-Velez, Byron C Wallace, and Ryan Adams. Graph-sparse lda: A topic model with structured sparsity. In *AAAI Conference on Artificial Intelligence*, pages 2575–2581, 2015.
- Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM, 2008.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- Katrin Erk and Sebastian Padó. A structured vector space model for word meaning in context. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 897–906, 2008.
- Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

- Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*, 2015.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.
- David Graff and C Cieri. English gigaword corpus. *Linguistic Data Consortium*, 2003.
- Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.
- Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2017.
- Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- Julian PT Higgins, Douglas G Altman, Peter C Gøtzsche, Peter Jüni, David Moher, Andrew D Oxman, Jelena Savović, Kenneth F Schulz, Laura Weeks, and Jonathan AC Sterne. The cochrane collaborations tool for assessing risk of bias in randomised trials. *Bmj*, 343:d5928, 2011.
- Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

- Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *arXiv preprint arXiv:1801.06146*, 2018.
- Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339, 2018.
- Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.
- Xinyu Hua and Lu Wang. A pilot study of domain adaptation effect for neural abstractive summarization. *arXiv preprint arXiv:1707.07062*, 2017.
- Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.
- Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558, 2016.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.
- Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.
- Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. Frustratingly easy neural domain adaptation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, December 2016.
- Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- Youngjoong Ko, Jinwoo Park, and Jungyun Seo. Automatic text categorization using the importance of sentences. In *Proceedings of the 19th international*

- conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.
- Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.
- Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*, 2019.
- Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of Association for Computational Linguistics*, pages 302–308, 2014.
- David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.
- Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.
- Lianghao Li, Xiaoming Jin, Sinno Jialin Pan, and Jian-Tao Sun. Multi-domain active learning for text classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1086–1094. ACM, 2012.
- Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.
- Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

- Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.
- Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.
- Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.
- Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. Optimization of image description metrics using policy gradient methods. In *International Conference on Computer Vision (ICCV)*, 2017.
- Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.
- Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.
- Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. Pre-trained multi-view word embedding using two-side neural network. In *Conference on Artificial Intelligence*, 2014.
- Gideon S Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *The Journal of Machine Learning Research*, 11:955–984, 2010.
- Iain J Marshall, Joël Kuiper, and Byron C Wallace. Automating risk of bias assessment for clinical trials. *Biomedical and Health Informatics, IEEE Journal of*, 19(4):1406–1412, 2015.
- Iain J Marshall, Joël Kuiper, and Byron C Wallace. Robotreviewer: evaluation of a system for automatically assessing bias in clinical trials. *Journal of the American Medical Informatics Association*, 23(1):193–201, 2016.
- James Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- SPFGH Moen and Tapio Salakoski² Sophia Ananiadou. Distributional semantics resources for biomedical text processing, 2013.
- Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 2315–2325, 2015.
- Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111*, 2016.
- Masaki Murata, Qing Ma, Kiyotaka Uchimoto, Hiromi Ozaku, Masao Utiyama, and Hitoshi Isahara. Japanese probabilistic information retrieval using location and category information. In *Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, pages 81–88. ACM, 2000.
- Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.
- Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics, 2012.
- Preksha Nema, Mitesh Khapra, Anirban Laha, and Balaraman Ravindran. Diversity driven attention model for query-based abstractive summarization. *arXiv preprint arXiv:1704.08300*, 2017.
- Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain J Marshall, Ani Nenkova, and Byron C Wallace. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, page 197. NIH Public Access, 2018.

- Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, et al. Neural information retrieval: At the end of the early years. *Information Retrieval Journal*, 21(2-3):111–182, 2018.
- Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.
- Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of Association for Computational Linguistics*, page 271, 2004.
- Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of the ACL*, 2005.
- Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.
- Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.
- Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.
- Nanyun Peng and Mark Dredze. Multi-task domain adaptation for sequence tagging. *ACL 2017*, page 91, 2017.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.
- Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. Distributional semantics resources for biomedical text processing. *Proceedings of Languages in Biology and Medicine*, 2013.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.
- Piyush Rai, Avishek Saha, Hal Daumé III, and Suresh Venkatasubramanian. Domain adaptation meets active learning. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 27–32. Association for Computational Linguistics, 2010.
- Maria E. Ramirez-Loaiza, Manali Sharma, Geet Kumar, and Mustafa Bilgic. Active learning: an empirical study of common baselines. *Data Mining and Knowledge Discovery*, pages 1–27, 2016.
- Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732, 2015.
- Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.
- Sebastian Ruder and Barbara Plank. Learning to select data for transfer learning with bayesian optimization. *arXiv preprint arXiv:1707.05246*, 2017.
- D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.
- Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*, 2017.

- Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2110–2118, 2016.
- Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.
- Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.
- Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478. Association for Computational Linguistics, 2011.
- Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.
- Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.
- Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.
- Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and SVN Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research*, 10(Nov):2615–2637, 2009.
- Aditya Siddhant and Zachary C Lipton. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*, 2018.
- Kevin Small, Byron Wallace, Thomas Trikalinos, and Carla E Brodley. The constrained weight space svm: learning with ranked features. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 865–872, 2011.
- Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic

- compositionality over a sentiment treebank. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
- Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, 2015.
- T. Terasawa, T. Dvorak, S. Ip, G. Raman, J. Lau, and T. A. Trikalinos. Charged Particle Radiation Therapy for Cancer: A Systematic Review. *Ann. Intern. Med.*, 2009.
- Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.
- Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.
- Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.
- Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.
- Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. Latent vector weighting for word meaning in context. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1012–1022, 2011.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

- Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.
- Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.
- Byron C Wallace, Thomas A Trikalinos, Joseph Lau, Carla Brodley, and Christopher H Schmid. Semi-automated screening of biomedical citations for systematic reviews. *BMC bioinformatics*, 11(1):55, 2010.
- Byron C. Wallace, Do Kook Choe, Laura Kertz, and Eugene Charniak. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of Association for Computational Linguistics*, pages 512–516, 2014.
- Byron C Wallace, Michael J Paul, Urmimala Sarkar, Thomas A Trikalinos, and Mark Dredze. A large-scale quantitative analysis of latent factors and sentiment in online doctor reviews. *Journal of the American Medical Informatics Association*, 21(6):1098–1103, 2014.
- Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.
- Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120, 2009.
- Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210, 2005.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.
- Fangzhao Wu, Yongfeng Huang, and Jun Yan. Active sentiment domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1701–1711, 2017.
- Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM, 2014.
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.
- Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*, 2017.
- Ainur Yessenalina, Yejin Choi, and Claire Cardie. Automatically generating annotator rationales to improve sentiment classification. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 336–341. Association for Computational Linguistics, 2010.
- Wenpeng Yin and Hinrich Schütze. Multichannel variable-size convolution for sentence classification. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 204–214, 2015.

- Dani Yogatama and Noah A Smith. Linguistic structured sparsity in text categorization. In *Meeting of the Association for Computational Linguistics*, pages 786–796, 2014.
- Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *ACL*, pages 545–550, 2014.
- Zhiguo Yu, Trevor Cohen, Elmer V Bernstam, and Byron C Wallace. Retrofitting word vectors of mesh terms to improve semantic similarity measures. *Intl. Workshop on Health Text Mining and Information Analysis at EMNLP*, pages 43–51, 2016.
- Omar Zaidan, Jason Eisner, and Christine D Piatko. Using” annotator rationales” to improve machine learning for text categorization. In *HLT-NAACL*, pages 260–267. Citeseer, 2007.
- Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012.
- Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*, 2017.
- Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.
- Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.
- Ye Zhang, Iain Marshall, and Byron C Wallace. Rationale-augmented convolutional neural networks for text classification. *arXiv preprint arXiv:1605.04469*, 2016.
- Ye Zhang, Stephen Roller, and Byron Wallace. Mgnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. *arXiv preprint arXiv:1603.00968*, 2016.
- Ye Zhang, Matthew Lease, and Byron C Wallace. Active discriminative text representation learning. In *AAAI*, pages 3386–3392, 2017.

- Ye Zhang, Matthew Lease, and Byron C Wallace. Exploiting domain knowledge via grouped weight sharing with application to text categorization. *arXiv preprint arXiv:1702.02535*, 2017.
- Ye Zhang, Nan Ding, and Radu Soricut. Shaped: Shared-private encoder-decoder for text style adaptation. *arXiv preprint arXiv:1804.04093*, 2018.
- Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. Bi-transferring deep neural networks for domain adaptation. In *ACL (1)*, 2016.
- Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1137–1144. Association for Computational Linguistics, 2008.

Vita

Ye Zhang was born in Harbin, Heilongjiang, China in 1989. He obtained a Bachelor degree in Communication Engineering from Harbin Institute of Technology in 2012, and then obtained a Masters of Science in Electrical and Computer Engineering from Carnegie Mellon University. After that, he entered computer science PhD program at the University of Texas at Austin. He has been working on natural language processing and machine learning.

Permanent Address: yezhang1989@gmail.com

This dissertation was typeset with $\text{\LaTeX} 2_{\epsilon}$ ¹ by the author.

¹ $\text{\LaTeX} 2_{\epsilon}$ is an extension of \LaTeX . \LaTeX is a collection of macros for \TeX . \TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.